

2001

# Combinatorial designs with applications in cryptography

Ghulam Rasool Chaudhry  
*University of Wollongong*

---

## Recommended Citation

Chaudhry, Ghulam Rasool, Combinatorial designs with applications in cryptography, Doctor of Philosophy thesis, School of Information Technology and Computer Science, University of Wollongong, 2001. <http://ro.uow.edu.au/theses/2007>

## **NOTE**

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

## **UNIVERSITY OF WOLLONGONG**

### **COPYRIGHT WARNING**

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.



# Combinatorial Designs with Applications in Cryptography

A thesis submitted in fulfillment of the  
requirements for the award of the degree

**Doctor of Philosophy**

from

UNIVERSITY OF WOLLONGONG

by

**Ghulam Rasool Chaudhry**

School of Information Technology and Computer Science  
October 2001

© Copyright 2001

by

Ghulam Rasool Chaudhry

All Rights Reserved

*Dedicated to*

*my parents*

*&*

*wife and children*

# Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Ghulam Rasool Chaudhry  
October 1, 2001

# Abstract

---

Since the evolution of human culture, small groups of trustworthy individuals have always played an important role in making crucial decisions in all areas of life. In information based systems, cryptography supports group activity by offering a wide range of cryptographic operations which can only be successfully executed if a well-defined group of people agrees to cooperate. Most of the stronger modern cryptographic systems have been designed and constructed using mathematical functions.

This dissertation looks at the minimal structures of combinatorial designs and their possible uses in cryptographic schemes. These minimal structures can be used to reconstruct a combinatorial design that can be set as a secret. The first objective of this thesis is to study minimal structures of combinatorial designs, especially of Room squares and latin squares. Uniquely completable and critical sets in Room squares are studied. General constructions for uniquely completable sets of Room squares are given but results on minimal and maximal critical sets are empirical only because of the structure of Room squares. General constructions of uniquely completable and critical sets for five different types of back-circulant latin squares are also given. The terms nest, power, influence and strong box in critical sets of Room squares and back-circulant latin squares are introduced. Latin interchanges are used to construct critical sets in modified-two back-circulant latin squares containing odd order subsquares. Critical sets for modified-two back-circulant latin squares for all odd  $n \leq 25$  are given and a conjecture is made for the general result. This is to be noted that, in this thesis, only strongly uniquely completable and strong critical sets of Room squares and latin squares are studied.

The second objective is to use the minimal structures of combinatorial designs in cryptographic applications, particularly secret sharing schemes. This thesis proposes generalised, hierarchical, key management and perfect secret sharing schemes based on

critical sets of Room squares. It shows how cheating in secret sharing schemes can be detected and prevented using critical sets of Room squares.

The third objective is to study another combinatorial structure, Bhaskar Rao designs (BRDs), which can also be used in cryptographic functions, particularly perfect hashing functions. Some open problems of BRDs for block size 4 are solved. This thesis solves most of the cases of  $\text{BRD}(v, 5, \lambda)$  for  $\lambda = 4, 10, 20$  and some other values of  $\lambda$ . A few cases of  $\text{BRD}(v, 6, \lambda)$  are also solved.



# Thesis Related Publications

---

1. G. Chaudhry and J. Seberry, Minimal and maximal critical sets in Room squares. *Proceedings of 7th Australasian Workshop on Combinatorial Algorithms (AWOCA'96)*, Australia, (1996), 75–86.
2. G. Chaudhry and J. Seberry, Secret sharing schemes based on Room squares. *Combinatorics, Complexity and Logic, Proceedings of DMTCS'96*, Springer-Verlag Singapore (1996), 158–167.
3. G. Chaudhry and J. Seberry, Minimal critical set of a Room square of order 7. *Bulletin of the ICA*, **20** (1997), 90.
4. G. Hossein, J. Pieprzyk, G. Chaudhry and J. Seberry, How to prevent cheating in Pinch's scheme. *Electronics Letters*, **33** (1997) 1453–54.
5. G. Chaudhry and J. Seberry, Room squares: critical sets and their bounds. Presented at the *3rd International Conference on Comb. Math. and Comb. Computing (3ICCMCC'97)*, Melbourne, Australia, July 1997.
6. G. Chaudhry, H. Ghodosi and J. Seberry, Perfect secret sharing schemes from Room squares. *J. Comb. Math. and Comb. Computing (JCMCC)* Canada, **28** (1998), 55–61.
7. G. Chaudhry and J. Seberry, On the  $(10, 5, \lambda)$ -family of Bhaskar Rao designs. *Bulletin of the ICA*, **23** (1998) 83–87.
8. L. Fitina, J. Seberry and G. Chaudhry, Back-circulant latin square and the influence of a set. *Aust. J. of Combinatorics*, **20** (1999) 163–180.
9. G. Chaudhry and J. Seberry, Influence of entries in critical sets of Room squares. *Bulletin of the ICA*, **28** (2000), 67–74.

10. G. Chaudhry, M. Greig and J. Seberry, On the  $(v, 5, \lambda)$ -family of Bhaskar Rao designs. *J. Stat. Planning and inference (JSPI)* (to appear).
11. G. Chaudhry and J. Seberry, On uniquely completable sets in Room squares. *Bulletin of the ICA* (to be revised).
12. G. Chaudhry, J. Seberry and R. Peddada, Uniquely completable sets in latin squares with subsquares of half order. *Aust. J. of Combinatorics* (to be revised).
13. G. Chaudhry and J. Seberry, Critical sets in modified-two back-circulant latin squares, (in preparation).

# Acknowledgements

---

I would like to thank my supervisor Prof. Jennifer Seberry for the stimulating discussions, the invaluable advice and the constructive critical comments during the work on this thesis. I am also thankful for her financial support during the course of my studies.

I would like to thank Dr. Malcolm Greig for working with us on Bhaskar Rao designs and proving many general results on  $BRD(v, 5, \lambda)$ .

I would also like to thank my fellow students Lakoa Fitina and Raju Pedadda for many helpful discussions about uniquely completable and critical sets in latin and Room squares. I am also thankful to another fellow student Hossein Ghodosi for his valuable guidance on secret sharing schemes and working with us on two papers.

I would also like to thank my wife, Shamim, and children, Asim, Aamir and Atif for their love, support and patience during the long hours of my research.

# Contents

---

Abstract	v
Thesis Related Publications	vii
Acknowledgements	ix
1 Introduction	1
1.1 Description of chapters . . . . .	2
I Room Squares	4
2 Uniquely Completable and Critical Sets	5
2.1 Introduction . . . . .	5
2.1.1 Critical sets . . . . .	9
2.2 Critical sets when all empty cells are known . . . . .	13
2.2.1 Algorithms to construct critical sets . . . . .	14
2.2.2 Cardinality . . . . .	15
2.3 Critical sets when empty cells are unknown . . . . .	18
2.3.1 Algorithms to construct critical sets . . . . .	20
2.3.2 Cardinality . . . . .	22
2.3.3 Complexity . . . . .	24
2.4 Uniquely completable sets . . . . .	26
2.5 Conclusion . . . . .	30
3 Power, Influence and Strong Box	31
3.1 Introduction . . . . .	31
3.2 Power, Influence and Strong Box . . . . .	32
3.3 Secret Sharing . . . . .	36

3.4	Conclusion . . . . .	37
<b>4</b>	<b>Secret Sharing Schemes</b>	<b>38</b>
4.1	Introduction . . . . .	38
4.2	Secret sharing schemes using Room squares when all empty cells are known . . . . .	45
4.2.1	Key Management Scheme . . . . .	47
4.3	Generalised schemes using Room squares when empty cells are unknown	48
4.3.1	A Generalised Scheme . . . . .	49
4.3.2	Hierarchical Scheme . . . . .	51
4.3.3	Key Management Scheme . . . . .	53
4.3.4	Security of the schemes . . . . .	54
4.4	A Perfect Scheme . . . . .	54
4.4.1	The Scheme . . . . .	55
4.4.2	Security of the scheme . . . . .	57
4.5	Cheating prevention in secret sharing . . . . .	57
4.5.1	The Scheme . . . . .	58
4.6	Conclusion . . . . .	59
<b>II</b>	<b>Back-Circulant Latin Squares</b>	<b>61</b>
<b>5</b>	<b>Uniquely Completable and Critical Sets</b>	<b>62</b>
5.1	Introduction . . . . .	62
5.2	Critical sets of small standard form latin squares . . . . .	63
5.3	Three constructions using subsquares . . . . .	65
5.3.1	Two Back-Circulant [BC(2,n)] . . . . .	66
5.3.2	Back-Circulant - Forward-Circulant [BCFC(2,n)] . . . . .	67
5.3.3	Modified-two Back-Circulant [MBC(2,n)] . . . . .	69
5.3.4	Algorithm . . . . .	71
5.4	Critical sets of latin squares of order $2^t$ . . . . .	73
5.5	Critical sets of latin squares of order $(mn)$ . . . . .	77
<b>6</b>	<b>Nest, Influence and Latin Collection</b>	<b>80</b>
6.1	Introduction . . . . .	80
6.2	Some general results . . . . .	84
6.3	Latin Collections . . . . .	90

6.4 Influence of a set . . . . . 91

6.5 A hierarchy of influence . . . . . 97

7 Critical Sets in Modified-Two Back-Circulants 101

7.1 Introduction . . . . . 101

7.2 A family of latin interchanges . . . . . 106

7.3 A family of critical sets . . . . . 128

7.4 Conclusion . . . . . 131

III Bhaskar Rao Designs 132

8 Bhaskar Rao Designs of Block Size 5 133

8.1 Introduction . . . . . 133

8.2  $BRD(v, 3, \lambda)$  and  $BRD(v, 4, \lambda)$  . . . . . 137

8.3 General Constructions . . . . . 138

8.4 Signing Known Designs . . . . . 141

8.5  $BRD(v, 5, \lambda)$  . . . . . 146

8.6  $BRD(10, 5, \lambda)$  . . . . . 151

8.7  $BRD(v, 5, 4)$  . . . . . 152

8.8  $BRD(v, 5, 10)$  . . . . . 156

8.9 Construction of BRDs from SDSs . . . . . 161

9 Bhaskar Rao Designs of Block Size 6 167

9.1 Some Constructions of  $BRD(v, 6, \lambda)$  . . . . . 167

9.2 Conclusion . . . . . 171

10 Concluding Remarks and Future Work 172

Bibliography 175

A Room Squares 184

A.1 Examples of Room squares . . . . . 184

A.1.1 Room squares  $R9_1 \dots R9_6$  listed in Tables 2.1 and 2.2 . . . . . 184

A.1.2 Room squares  $R11_1 \dots R11_3$  listed in Table 2.1 . . . . . 184

A.1.3 Room squares  $R11_4 \dots R11_6$  listed in Table 2.2 . . . . . 185

A.2 Examples of critical sets . . . . . 186

A.2.1 Minimal and maximal critical sets listed in Table 2.1 . . . . . 186

A.2.2	Minimal and maximal critical sets listed in Table 2.2 . . . . .	187
A.3	Completion of a partial Room square . . . . .	196
A.4	Completion of a partial Room square . . . . .	197
<b>B</b>	<b>Latin Squares</b>	<b>198</b>
B.1	Examples of critical sets for $BC(2, n)$ . . . . .	198
B.2	Examples of critical sets for $BCFC(2, n)$ . . . . .	202
B.3	Examples of critical sets for $MBC(2, n)$ . . . . .	204
B.4	General constructions of uniquely completable and critical sets . . . . .	208
B.5	Latin Interchanges in $MBC(2, n)$ . . . . .	214
<b>C</b>	<b>Bhaskar Rao Designs</b>	<b>215</b>
C.1	Thirteen inequivalent $BRD(10, 5, 4)$ 's . . . . .	215
<b>D</b>	<b>Source Code for Room Squares, Latin Squares and BRDs</b>	<b>219</b>

# List of Figures

---

4.1	A projective space . . . . .	39
4.2	A polynomial of degree 2 . . . . .	43
7.1	Sub-division of partial subsquare AR for $m < 4$ . . . . .	107
7.2	Sub-division of partial subsquare AR for $m \geq 4$ . . . . .	107
7.3	Coordinates of the partial subsquare S1 . . . . .	108
7.4	Coordinates of the partial subsquare S2 . . . . .	108
7.5	Coordinates of the partial subsquare S3 . . . . .	108
7.6	Coordinates of the partial subsquare S4 . . . . .	109
7.7	Coordinates of the partial subsquare S5 . . . . .	109



# Chapter 1

---

## Introduction

Cryptography is the science that deals with the design of algorithms, protocols and systems for solving two kinds of security problems: *privacy* and *authentication*. More precisely, cryptography is the use of transformations of data intended to make the data useless to opponents, but meaningful to legitimate receivers. The only secret Part of almost all modern cryptographic systems, however, is the *key* – the parameter that selects the particular transformation to be employed. So there is a clear need to provide the secrecy to such sensitive information.

Groups play an important role in the modern world. Cryptography supports group activity by offering a wide range of cryptographic operations which can only be successfully executed if a well defined group of people agrees to cooperate. One of the areas of cryptography, secret sharing schemes provide the secrecy to sensitive information by partitioning it into several parts in such a way that a specified number of the parts must be combined in order to recover the original information, the secret. So losing a piece does not compromise the secret, that is, the opponent cannot learn the secret as long as he does not have access to a predetermined number of pieces. Examples of such information include the code to activate a nuclear weapon, open a bank vault, cryptographic master keys in a Key Distribution Centre (KDC), proprietary trade-secret formulae, etc.

Secret sharing schemes were first introduced by Blakley [18], Shamir [98] and Chaum [32] in 1979, and subsequently have been studied by numerous other authors, for a general discussion of shared secret schemes, see the Simmons paper [100]. A number of mathematical structures have been used to model shared secret schemes. Some of these are polynomials, geometric configurations, block designs, Reed-Solomon codes, vector spaces, matroids, near-right fields, complete multipartite graphs, orthogonal arrays and back-circulant latin squares.

This research has been motivated by studies of secret sharing schemes by Cooper, Donovan and Seberry [38], particularly as they lend themselves easily to hierarchical

and compartmentalised secret sharing, key distribution schemes by Merkle [84], situations where receipt of the message is delayed by only providing partial information and requiring the receiver to take time to solve the puzzle, situations where the sender or the communication channel is limited to a small size and some problems in the design of experiments. Critical sets of latin squares have been studied by various authors, e.g., Nelder [87], Curran and van Rees [40], Smetaniuk [101], Stinson and van Rees [103], Cooper, Donovan and Seberry [37] and Gower [60].

In this thesis, minimal structures of combinatorial designs and their properties are studied which can be used in secret sharing schemes and other cryptographic applications. Uniquely completable and critical sets in Room squares are introduced and secret sharing schemes based on critical sets of Room squares are proposed. Uniquely completable and critical sets for different types of back-circulant latin squares are studied. The terms power, influence and strong box in critical sets of Room squares are introduced which can specifically be used in cryptographic applications. The terms nest, influence, latin collection and strong box in critical sets of latin squares are also introduced. Latin interchanges are used to construct critical sets in modified-two back-circulant latin squares.

Another combinatorial structure, Bhaskar Rao designs (BRD) have also useful application in cryptographic functions, particularly perfect hashing functions. Bhaskar Rao designs were invented by Bhaskar Rao in 1966 in his papers [16, 17] and since then have been studied by number of authors: de Launey [41], de Launey and Sarvate [42], de Launey and Seberry [45, 46], de Launey, Sarvate and Seberry [44], Gibbons and Mathon [59], Lam and Seberry [78], Palmer and Seberry [90], Seberry [95, 96], Singh [99], Street [106], Street and Rodger [107], and Vyas [110]. BRDs with block size 3 were studied by Singh [99], Vyas [110], and Seberry [95, 96]. BRDs with block size 4 were studied by de Launey and Seberry [45, 46], see [43] for a recent survey. In this thesis, BRDs of block size 5 and 6 are studied.

## 1.1 Description of chapters

This thesis consists of three Parts. In Part I, uniquely completable and critical sets of Room squares are studied. Secret sharing schemes based on critical sets of Room squares are proposed. The terms power, influence and strong box in Room squares are introduced. In Part II, uniquely completable and critical sets of different types of back-circulant latin squares are studied. The terms nest, influence, latin collection

and strong box are introduced in back-circulant latin squares. Latin interchanges are used to construct critical sets of modified-two back-circulant latin squares. In Part III, Bhaskar Rao Designs of block size 5 and 6 are constructed.

Part I of the thesis consists of Chapters 2, 3 and 4. In Chapter 2, the concepts of uniquely completable sets and critical sets in Room squares is introduced. Exhaustive search and hill-climbing algorithms are used to find bounds for minimal and maximal critical sets of Room squares. In Chapter 3, the terms power and influence of entries in the critical sets of Room squares are introduced. The term strong box in Room squares is also defined. In Chapter 4, new secret sharing schemes based on critical sets of Room squares are proposed. Generalized, hierarchical, key management and perfect schemes arising from critical sets of Room squares are also proposed. It is also shown how cheating can be detected using schemes based on Room squares.

Part II of the thesis consists of Chapters 5, 6 and 7. In Chapter 5, three different back-circulant latin squares are studied and their uniquely completable and critical sets are constructed. A few constructions of uniquely completable and critical sets of latin squares of orders  $2^t$  and  $mn$  are also given. In Chapter 6, the terms nest, latin collection and influence of sets in the critical sets of back-circulant latin squares are introduced and studied. The size of the strong box in back-circulant latin squares is also studied. In Chapter 7, a theorem for the construction of a family of critical sets in modified-two back-circulant latin squares  $MBC(2, 2m+1)$  is proved using latin interchanges.

Part III of the thesis consists of Chapters 8 and 9. In Chapter 8, Bhaskar Rao designs of block size 5 are constructed. General constructions of BRDs of block size 5 are also given for different orders of  $\lambda$ . In Chapter 9, Some constructions of Bhaskar Rao designs of block size 6 are given.

Finally, in Chapter 10, concluding remarks are given and some directions for future research in the areas of combinatorial designs and secret sharing schemes are also highlighted. In the appendices, examples of critical sets of Room squares and latin squares, examples of latin interchanges for modified-two back-circulant latin squares and examples of  $BRD(v, 5, \lambda)$ 's are given. Source code to construct uniquely completable and critical sets in latin squares and Room squares is attached at the end of the thesis. Source code for the construction of Bhaskar Rao designs is also attached.

# Part I

## Room Squares

# Chapter 2

---

## Uniquely Completable and Critical Sets

Section 2.2 is based on a paper by Chaudhry and Seberry [22]. More than 95 percent of the results are due to the first author. Section 2.3 is based on the papers by Chaudhry and Seberry [24, 25]. More than 95 percent of the results are due to the first author. Overall about 95 percent of the work in this chapter is due to the author of this thesis.

### 2.1 Introduction

Many authors have studied the minimum amount of information needed to recreate combinatorial structures. Critical sets in latin squares have been studied by Nelder [87], Curran and van Rees [40], Smetaniuk [101], Stinson and van Rees [103], Cooper, Donovan and Seberry [37], Gower [60], Cooper, Donovan and Gower [39] and Khodkar [74]. Minimum defining sets of combinatorial designs, see for example [106], have been studied by Street, Sarvate, Kunkle, Seberry et al.. Critical sets have a number of applications in both agriculture and cryptography. This research has been motivated by studies of secret sharing schemes by Cooper et al., key distribution schemes by Merkle (PhD thesis), situations where we wish to delay a receipt of the message by only providing partial information and require the receiver to take time to solve the puzzle, situations where the sender or the communication channel is limited to a small size and some problems in the design of experiments. In this chapter, we widen the structures considered to include uniquely completable and critical sets in Room squares.

Room squares were first introduced by R. R. Anstice in 1852 [10, 11] and then by T. G. Room in 1955 [93]. Room squares are important in the organisation of the bridge tournaments, in paired-comparison experiments and round-robin tournaments. Suppose there are eight teams in a football tournament. Each team has to play with other teams exactly once in seven days. There are seven referees. Each referee should have each team exactly once, to eliminate bias. A room square of order 7 solves this

problem.

To find the true minimum information needed to reconstruct a Room square, we should have the minimum possible structure imposed on the Room square. However, for comparative purposes, it is valuable to compare the size of the least critical set under a number of scenarios.

A *Room square*  $R$  of order  $r$  is an  $r \times r$  array each of whose cells may either be empty or contain an unordered pair of objects  $0, 1, 2, \dots, r$ , subject to the following conditions:

- (i) each of the objects  $0, 1, 2, \dots, r$  occurs precisely once in each row of  $R$  and precisely once in each column of  $R$ , and
- (ii) every possible unordered pair of objects occurs precisely once in the whole array.

A *partial Room square*  $P$  of order  $r$  is an  $r \times r$  array each of whose cells may either be empty or contain an unordered pair of objects  $0, 1, 2, \dots, r$ , subject to the following conditions:

- (i) each of the objects  $0, 1, 2, \dots, r$  occurs at most once in each row of  $R$  and at most once in each column of  $R$ , and
- (ii) every possible unordered pair of objects occurs at most once in the whole array.

**Theorem 2.1** (Mullin and Wallis [85]) *There exists a Room square of every odd integer order  $r$  greater than or equal to 7.*

A graph  $G = G(V, E)$  consists of a finite set  $V$  of objects called *vertices* and a set  $E$  of unordered pairs of members of  $V$ , called *edges*. A *one-factor* in a graph  $G$  is a set of edges in which every vertex appears precisely once. A *one-factorisation* of  $G$  is a set of one-factors which has no common edge but between them contain every edge of the graph. In other words, it is a partition of the edge-set of  $G$  into one-factors. The *Order* of a one-factorisation of complete graph on  $n + 1$  vertices is  $n$ . One-factorisations of complete graphs arise naturally in the construction of round-robin tournaments and other competition schedules. One-factorisations were first introduced by Kirkman [75] in 1847.

**Theorem 2.2** (Wallis [113]) *One-factorisations of  $K_{n,n}$  are equivalent to latin squares of order  $n$ .*

Two one-factorisations  $F$  and  $H$  of  $G$ , say  $F = \{f_1, f_2, \dots, f_k\}$ ,  $H = \{h_1, h_2, \dots, h_k\}$ , are called *isomorphic* if there exists a map  $\phi$  from the vertex-set of  $G$  onto itself such that  $\{f_1\phi, f_2\phi, \dots, f_k\phi\} = \{h_1\phi, h_2\phi, \dots, h_k\phi\}$ .

Two one factors are called *orthogonal* if their intersection consists of at the most one edge; two one-factorisations are *orthogonal* if each factor in the first is orthogonal to each factor in the second. A pair of orthogonal one-factorisations of a complete graph on  $n + 1$  vertices is equivalent to a Room square of order  $n$  for  $m \geq 3$  and  $n = 2m + 1$ .

**Example 2.1** Two orthogonal one-factorisations  $F_1$  and  $F_2$  of order 7 are given below respectively:

01	27	36	45	01	24	37	56
02	13	47	56	02	14	35	67
03	15	24	67	03	17	25	46
04	17	26	35	04	12	36	57
05	12	37	46	05	16	23	47
06	14	23	57	06	15	27	34
07	16	25	34	07	13	26	45

Set up a  $2m + 1$  by  $2m + 1$  *Room square* as follows: if the  $i$ th one-factor in  $F_1$  and the  $j$ th one-factor in  $F_2$  have the edge  $[x, y]$  in common , then place  $[x, y]$  in the cell  $(i, j)$  of the array. If those two factors have no common edge, leave the cell empty.

**Theorem 2.3 (Dinitz and Stinson [51])** *A Room square of order  $r$  is equivalent to a pair of orthogonal one-factorisations of a complete graph.*

We use the most effective *hill-climbing algorithm*, given by Dinitz and Stinson in [50] to construct examples of one-factorisations and consequently Room squares of different orders. Computer programs have been written in C++ and are attached in Appendix D. Room squares can also be constructed using orthogonal starters and some other methods, but we will not discuss them in this thesis.

Denote  $N_r = 0, 1, 2, \dots, r$ . A Room square of order  $r$  based on  $N_r$  is *standardised* if the  $i$ th diagonal cell, cell  $(i, i)$ , contains  $\{0, y\}$  for  $1 \leq i, y \leq r$ .

**Theorem 2.4 (Wallis [112])** *If there is a Room square of side  $r$ , then there is a standardised Room square of order  $r$ .*

A *skew* Room square  $R$  [111] is a Room square with the property that when  $i \neq j$ , either the  $(i, j)$  or the  $(j, i)$  cell of  $R$  is occupied, but not both.

**Theorem 2.5 (Dinitz and Stinson [51])** *There is a skew Room square of order  $r$ , if and only if  $r$  is odd and  $r \neq 3$  or  $5$ .*

**Example 2.2** A standardised and skew Room square of order 7:

0 1	-	4 5	6 7	-	-	2 3
5 7	0 2	-	-	-	1 3	4 6
-	5 6	0 3	1 2	-	4 7	-
-	3 7	-	0 4	2 6	-	1 5
3 6	1 4	2 7	-	0 5	-	-
2 4	-	-	3 5	1 7	0 6	-
-	-	1 6	-	3 4	2 5	0 7

**Example 2.3** A standardised and skew Room square of order 9:

0 1	-	4 9	3 7	2 8	-	5 6	-	-
8 9	0 2	-	-	-	5 7	3 4	-	1 6
-	5 8	0 3	-	6 9	2 4	-	1 7	-
-	3 6	7 8	0 4	-	1 9	-	2 5	-
-	7 9	-	1 2	0 5	3 8	-	4 6	-
4 5	-	-	-	-	0 6	1 8	3 9	2 7
-	-	2 6	5 9	1 3	-	0 7	-	4 8
6 7	1 4	-	-	-	-	2 9	0 8	3 5
2 3	-	1 5	6 8	4 7	-	-	-	0 9

Two Room squares  $R$  and  $S$  are said to be *isomorphic* [112] if  $R$  can be obtained from  $S$  by permuting the rows and columns and relabelling the elements. Two Room squares  $R$  and  $S$  are *equivalent* if  $R$  is isomorphic to  $S$  or to the transpose of  $S$ .

**Theorem 2.6 (Wallis [112])** *If  $NR(n)$  is the number of non-isomorphic Room squares of order  $n$  and  $IR(n)$  is the number of inequivalent Room squares of order  $n$ , then  $NR(n) \leq 2IR(n)$*

There are exactly 10 non-isomorphic Room squares of order 7 and 511,562 non-isomorphic Room squares of order 9 given by Dinitz in Section 4.39.10. of [34]. All inequivalent Room squares of order 7 and 9 have been found from one-factorisations of a complete graph. There are exactly 6 inequivalent Room squares of order 7, see [63, 64, 112] and 257,630 Room squares for order 9, see [12, 51, 52]. No exact number is known for higher order Room squares. Although there are:



526,915,620 non-isomorphic one-factorisations of order 11 ( $K_{12}$ )

$9.876 \times 10^{28}$  distinct one-factorisations of order 13 ( $K_{14}$ )

$1.148 \times 10^{44}$  distinct one-factorisations of order 15 ( $K_{16}$ )

$1.520 \times 10^{63}$  distinct one-factorisations of order 17 ( $K_{18}$ )

and the number of possible Room squares for each of these orders should be even larger, see [49] for further details.

In a Room square  $R$  of order  $r$ , a set of  $s$  rows and  $s$  columns said to form a Room subsquare  $S$  of order  $s$  if the  $s \times s$  array formed by their intersection is itself precisely a Room square based on some  $s+1$  of the elements.

**Theorem 2.7 (Wallis [111])** *If there is a Room square of order  $r$  with a subsquare of order  $s$ , where  $s < r$ , then  $r \geq 3s + 2$*

### 2.1.1 Critical sets

A number of possible scenarios for *critical sets* in Room squares considered are as follows:

1. The order of the Room square and all the empty cells are given. A critical set consists of a set of quadruples  $(i, j; k \ l)$  which indicates that the pair of integers  $(k \ l)$  occupies the cell  $(i, j)$ .
2. The order of the Room square is given. A critical set consists of a set of quadruples  $(i, j; k \ l)$  which indicates that the pair of integers  $(k \ l)$  occupies the cell  $(i, j)$  and triples  $(i, j; -)$  which indicates that cell  $(i, j)$  is empty.
3. The order of the Room square is given. A critical set consists of a set of quadruples  $(i, j; k \ l)$  which indicates that the pair of integers  $(k \ l)$  occupies the cell  $(i, j)$ .
4. The order of the Room square and all the empty cells are given. A critical set consists of a set of quadruples  $(i, j; k \ l)$  which indicates that the pair of integers  $(k \ l)$  occupies the cell  $(i, j)$  and triples  $(i, j; z)$  which indicates that the integer  $z$  occupies the cell  $(i, j)$ .
5. The order of the Room square is given. A critical set consists of a set of quadruples  $(i, j; k \ l)$  which indicates that the pair of integers  $(k \ l)$  occupies the cell  $(i, j)$ , triples  $(i, j; z)$  which indicates that the integer  $z$  occupies the cell  $(i, j)$  and triples  $(i, j; -)$  which indicates that cell  $(i, j)$  is empty.

6. The order of the Room square is given. A critical set consists of a set of quadruples  $(i, j; k \ l)$  which indicates that the pair of integers  $(k \ l)$  occupies the cell  $(i, j)$ , triples  $(i, j; z)$  which indicates that the integer  $z$  occupies the cell  $(i, j)$ .

In all cases the information given by the critical set should allow the Room square to be uniquely completed. In all cases the removal of any of the information in the critical set should allow more than one Room square to be formed from the remaining information.

From the perspective of secret sharing, certain scenarios have advantage of keeping more information confidential. For example, scenario 1 which gives all the empty cells. In cryptographic applications, it is common to give out lots of information but not enough for an enemy to deduce any secret information.

Giving the positions of all the empty cells gives a lot of information. The integer  $n$  can be described by  $\lceil \log_2 n \rceil + 1$  bits. Any Room square of order  $n$  contains  $\binom{n(n+1)}{2}$  pairs of integers in  $\binom{n(n+1)}{2}$  cells, so it is completely described by  $\binom{n+1}{2}$  quadruples  $(i, j; k \ l)$  which gives the pair  $(k \ l)$  in the cell  $(i, j)$ . This takes  $R(n)$  bits where

$$\begin{aligned} R(n) &= \text{number of cells} \times \text{bits need to describe a cell plus its contents} \\ &= \frac{n(n+1)}{2} 4(\lceil \log_2 n \rceil + 1) \\ &= 2n(n+1)(\lceil \log_2 n \rceil + 1) \end{aligned}$$

A triple which describes an empty cell also gives information, in fact  $3(\lceil \log_2 n \rceil + 1)$  bits. There are a total of  $\frac{n(n-1)}{2}$  empty cells. Hence in fact knowing all the empty cells gives  $\frac{3}{2}n(n-1)(\lceil \log_2 n \rceil + 1)$  bits of information.

Here are some other comments:

1. Giving out information can be disastrous in cryptographic applications.
2. There is no reason why empty cells can not be treated as two element or one element cells.
3. When completing a Room square, often one can reason that a certain element is in a certain cell and its cell-mate is deduced separately. So one element sets are just as natural as two element sets for a critical set.
4. Scenarios 2 and 3 are similar to the definition of critical sets in latin squares.
5. Counting elements in the critical set is what really happens in latin squares so scenario 6 is most like the definition of critical sets in latin squares.

6. Scenario 5 gives the most flexibility.

A *uniquely completable set* is a partial Room square which completes uniquely to a Room square but may be able to have some elements removed to form a critical set. If a Room square  $R$  contains a  $s \times s$  subarray  $S$  and if  $S$  is a Room square of order  $s$ , then it is said that  $S$  is a *Room subsquare* of  $R$ .

Generally speaking: a *critical set*  $Q = \{Q_1, Q_2, Q_3, \dots, Q_c\}$ ,  $|Q| = c$ , in a Room square  $R$  of order  $r$ , is a set of quadruples  $[i, j; k \ l]$  and /or triples  $[i, j; -]$  such that if any quadruple or triple is removed from the set, it can no longer be uniquely completed, see [101, 40]. In a quadruple  $[i, j; k \ l]$ , the cell  $(i, j)$  is the position of a pair of integers  $(k \ l)$  in the square. In a triple  $[i, j; -]$ , the cell  $(i, j)$  is the position of the empty cell  $(-)$  in the square.  $Q$  provides minimal structure from which  $R$  can be reconstructed uniquely.

In latin squares, any subset of a critical set will have two or more completions. But critical sets of Room squares are slightly different from critical sets in latin squares because of the structure of Room squares and conditions for their strong completion. Any subset of a critical set of a Room square may have two or more strong completions or it may have one strong completion and for all other possible combinations of elements, there is no strong completion. This does not preclude the possibility that there is smaller critical set which has a weak completion. We have not studied weak completions in this thesis. Let us consider the following critical set of the Room square given in Example 2.5:

		4 5	6 7			
	0 2				1 3	
	5 6		1 2			
		2 7				
				1 7		
				3 4		0 7

A complete computer search was made by removing the contents of each cell in turn from the critical set. Two cases were found:

- when the quadruple  $(3,4; 1 \ 2)$  was removed, there were two completions as given below:

0 1	-	4 5	6 7	-	-	2 3
5 7	0 2	-	-	-	1 3	4 6
-	5 6	0 3	1 2	-	4 7	-
-	3 7	-	0 4	2 6	-	1 5
3 6	1 4	2 7	-	0 5	-	-
2 4	-	-	3 5	1 7	0 6	-
-	-	1 6	-	3 4	2 5	0 7

0 3	-	4 5	6 7	-	-	1 2
5 7	0 2	-	-	-	1 3	4 6
-	5 6	0 1	2 3	-	4 7	-
1 6	3 7	-	0 4	2 5	-	-
-	1 4	2 7	-	0 6	-	3 5
2 4	-	3 6	-	1 7	0 5	-
-	-	-	1 5	3 4	2 6	0 7

- when any other quadruple was removed from the critical set, no other possible completion could be found by the exhaustive search. The data for this search is given in Appendix A.4.

The set  $S$  is said to be a *strong critical set* of a Room square  $R$  of order  $r$  if there exists a set  $\{P_1, P_2, P_3, \dots, P_m\}$  of  $m = r^2 - |S|$  partial Room squares, of order  $r$ , such that:

- (1)  $S \subseteq P_1 \subseteq P_2 \subseteq \dots P_m \subseteq R$ .
- (2) for any  $s$ , with  $2 \leq s \leq m$ , where  $P_s = P_{s-1} \cup \{(i, j; k \ l)\}$ , the set  $P_{s-1} \cup \{(i, j; k' \ l')\}$  is not a partial Room square for any pair  $(k' \ l') \in N_r \setminus (k \ l)$ , where  $N_r = 0, 1, 2, 3, \dots r$ .

A critical set is said to be *non-deterministic* if it is not strong: that is, the partial Room square has no forced quadruples but it is still uniquely completable. Non-deterministic critical sets are much harder to find as they need an exhaustive search.

A *minimal critical set* (*min. cs*) of a Room square  $R$  of order  $r$  is a critical set of minimum cardinality whereas a *maximal critical set* (*max. cs*) is a critical set of maximum cardinality.

$Q^s$  is the *smallest critical set* (*scs*) if  $|Q^s| = s$  is minimum of all critical sets of the Room squares of order  $r$  whereas  $Q^l$  is the *largest critical set* (*lcs*) if  $|Q^l| = l$  is the largest of all critical sets of the Room squares of order  $r$ , see [37, 40, 103]. Hence a *min. cs* is the smallest of all the critical sets of a particular Room square of order  $r$  whereas *scs* is the smallest of all the critical sets of all the inequivalent Room squares of order  $r$ . Similarly a *max. cs* is the largest of all the critical sets of a particular Room square of order  $r$  whereas *lcs* is the largest of all the critical sets of all the inequivalent Room squares of order  $r$ .

Throughout this thesis, we will be working on strong critical sets only. So we will use the term “critical set” to represent the “strong critical set”. The two reasons we only consider strong completions are:

1. We have not yet discovered any example with a weak completion.
2. We have not been able to find an analogous concept to that of latin interchanges for Room squares as by experiment.

Also min. cs will represent minimal strong critical set and max. cs represents a maximal strong critical set. Every cell of a Room square of order  $r$  is either empty, a triple  $(i, j; -)$ , or contains a pair of integers, a quadruple  $(i, j; k \ l)$  where  $i, j, k$  and  $l$  range from  $0, 1, \dots, r$ . Call a triple or a quadruple as *an entry*.

In this chapter, the scope of studying the critical sets is limited to the first two scenarios given in the beginning of this section. We will also investigate their role in secret sharing. These scenarios can be stated as: 1) Critical sets when the order of Room square and positions of all empty cells are known and 2) Critical sets when only the order of Room square is known.

## 2.2 Critical sets when all empty cells are known

Given the order of the Room square and positions of all the empty cells (triples). A *critical set*  $Q = [Q_1, Q_2, Q_3, \dots, Q_c]$ ,  $|Q| = c$ , in a Room square  $R$  of order  $r$ , is a set of quadruples  $Q_a = [i, j; k \ l]$  such that if any  $Q_a$  is removed from the set, it can no longer be uniquely completed.

### Completion Rules

Four rules are given which will be used in completing a Room square. A Room square  $R$  of order  $r$  is written using  $r + 1$  elements and  $\frac{r-1}{2}$  empty cells in each row/column of  $R$ :

1. A Room square  $R$  of order  $r$  is written using  $r + 1$  elements. If any row  $i$  and column  $j$  have  $r - 1$  elements already known, then if the  $(i, j)$ th entry is non-empty, it is known.
2. A Room square  $R$  of order  $r$  is written using  $r + 1$  elements. If in row  $i$  and column  $j$ ,  $(r + 1 - k)$  elements are known and  $\binom{k}{2} - 1$  of the pairs, on the  $k$  unknown entries, already exist in the square, then if the  $(i, j)$ th entry is non-empty, it is known.

3. If after deleting a pair of integers from a Room square, we are still able to uniquely complete that square, then that entry is not a member of the critical set.

### 2.2.1 Algorithms to construct critical sets

*Algorithm to construct a critical set:*

Suppose  $R$  is the original Room square of order  $r$  and positions of all its empty cells are known. Take a copy  $R'$  of a Room square of order  $r$  corresponding to  $R$ . To find a critical set  $Q$ , take an empty square  $Q$  with empty cells corresponding to  $R'$ . Select a quadruple, i.e., a pair of integers  $(k \ l)$  in the cell  $(i, j)$ , at random from  $R'$ , delete it for the time being and try to complete  $R'$  using completion rules given above. If  $R'$  is completed without that quadruple, then delete that quadruple from  $R'$  permanently; otherwise place it back into  $R'$  and place it in  $Q$  as well (thus preventing its reuse). Repeat this process of temporary deletion until none of the remaining quadruples (pairs) in  $R'$  can be deleted. Hence  $Q$  is the *critical set* of that square, since at this stage  $Q$  equals  $R'$  and no further quadruples can be deleted from  $R'$ .

*Algorithm to construct a min. cs:*

1. Let  $Q$  be a critical set of a Room square  $R$  of order  $r$ , we consider all possible combinations each consisting of  $|Q| - 1$  quadruples from the Room square.
2. for each partial Room square consisting of  $|Q| - 1$  quadruples:
3. Try to complete the partial square using completion rules.
4. If any of the above partial squares is uniquely completed, check if it is critical. If it is, then  $Q - 1$  is the new critical set. Replace the original  $Q$  set by the new set containing  $|Q| - 1$  quadruples, rename it as  $Q$  and now repeat the process from step 1.
5. If none can be completed, then the minimal critical set for that square is the current  $Q$ .

*Algorithm to construct a max. cs:*

1. Let  $Q$  be a critical set of a Room square  $R$  of order  $r$ , we consider all possible combinations each consisting of  $|Q| + 1$  quadruples from the Room square.
2. for each set of  $|Q| + 1$  quadruples, perform the following steps:

3. for each partial square consisting of  $|Q|$  quadruples from the  $|Q| + 1$  quadruples in step (2), perform the following steps:
  4. Try to complete the partial square using the completion rules.
  5. If the square is not uniquely completed, then repeat steps 3 and 4 otherwise discard the set containing  $|Q| + 1$  quadruples, and go to step 2 for next set of  $|Q| + 1$  quadruples.
  6. If none of the  $Q$  subsets can be completed, then try to complete the partial square consisting of  $|Q| + 1$  quadruples in step 3.
7. If any set of  $|Q| + 1$  quadruples complete the square uniquely, then check whether it is critical; if it is, then  $Q+1$  is the new critical set for that square, replace the original  $Q$  set by the new set containing  $|Q| + 1$  quadruples, rename it as  $Q$  and now repeat the perform from step 1.
8. If none of partial squares consisting of  $|Q| + 1$  quadruples is a critical set, then the maximal critical set for that square is the current  $Q$ .

### 2.2.2 Cardinality

The bounds for minimal (*min. cs*) and maximal (*max. cs*) critical sets have been computed for all inequivalence classes of Room squares of order 7 and some for order 9 and 11. Examples of *min. cs* and *max. cs* are given in Appendix A.2. The results are summarised in Table 2.1 below. In this table, all the Room squares of order 7 have been taken from [112] whereas the Room squares of order 9 and 11 were generated using hill-climbing algorithms given in [50] and are listed in Appendix A. Bounds for order 9 and 11, given in this table, are as per results obtained because an exhaustive search was not possible, actual minimal critical sets may be smaller than the ones given and maximal critical sets may be much larger than the ones given.

Table 2.1: Cardinality of *min. cs* and *max. cs*

Order of Room square	Class Representative	<i>Cardinality</i>	
		max. cs	min. cs
7	R11	10	8
	R14	10	8
	R15	10	8
	R16	10	8
	R44	10	8
	R45	10	8
7 Skew	R11	10	8
	R15 $\tilde{6}$	10	8
	R45 $\tilde{5}$	10	8
9	R9 <sub>1</sub>	17	15
	R9 <sub>2</sub>	16	13
	R9 <sub>3</sub>	16	14
	R9 <sub>4</sub>	15	14
	R9 <sub>5</sub>	14	12
	R9 <sub>6</sub>	16	15
9 Skew	R9 <sub>1</sub>	17	15
11	R11 <sub>1</sub>	24	20
	R11 <sub>2</sub>	26	22
	R11 <sub>3</sub>	25	21

**Example 2.4** A Room square of order 7 and one of its critical sets.

0 1	-	4 5	6 7	-	-	2 3
5 7	0 2	-	-	-	1 3	4 6
-	5 6	0 3	1 2	-	4 7	-
-	3 7	-	0 4	2 6	-	1 5
3 6	1 4	2 7	-	0 5	-	-
2 4	-	-	3 5	1 7	0 6	-
-	-	1 6	-	3 4	2 5	0 7



*	-	4 5	*	-	-	2 3
*	*	-	-	-	1 3	4 6
-	*	0 3	1 2	-	4 7	-
-	*	-	*	2 6	-	1 5
*	*	*	-	*	-	-
*	-	-	*	*	*	-
-	-	*	-	*	*	*

In the above square, “\*” shows the unknown pair cell and “-” shows empty cell. In this example, the order of the Room square is 7 and all empty cells are given. The critical set  $Q$  for the above Room square contains the following quadruples: (1,3;4 5), (1,7;2 3), (2,6;1 3), (2,7;4 6), (3,3;0 3), (3,4;1 2), (3,6;4 7), (4,5;2 6), (4,7;1 5). That is,  $Q$  is the only critical set of the Room square of order 7 with a pair (4 5) in the cell (1,3), pair (2 3) in the cell (1,7), pair (1 3) in the cell (2,6), pair (4 6) in the cell (2,7), pair (0 3) in the cell (3,3), pair (1 2) in the cell (3,4), pair (4 7) in the (3,6), pair (2 6) in the cell (4,5) and pair (1 5) in the (4,7). So if any quadruple is deleted, the Room square cannot be completed uniquely. It is noted that the information provided is 12 bits for each quadruple and 9 bits for each empty cell. So we have used  $9 \times 12 + 21 \times 9 = 297$  bits to describe the above critical set.

It is also noted that knowledge of all the empty cells is not always necessary to complete the Room square. In the following partial Room square, only 7 empty cells along with quadruples (pairs of integers) are enough to complete the above Room square which contains  $9 \times 12 + 7 \times 9 = 171$  bits of information:

*	*	4 5	*	*	*	2 3
*	*	*	-	*	1 3	4 6
*	*	0 3	1 2	*	4 7	*
*	*	*	*	2 6	*	1 5
*	*	*	-	*	*	*
*	-	-	*	*	*	*
-	-	*	-	*	*	*

But if we try to complete the Room square using quadruples only, the square cannot be completed uniquely. In the following square, only four empty cells could be filled and the square contains  $9 \times 12 + 4 \times 9 = 144$  bits of information:

*	*	4 5	*	*	*	2 3
*	*	*	*	*	1 3	4 6
*	*	0 3	1 2	-	4 7	-
*	*	-	*	2 6	-	1 5
*	*	*	*	*	*	*
*	*	*	*	*	*	*
*	*	*	*	*	*	*

It is also noted that even if all the empty cells are given, if one quadruple is deleted from the critical set, then the square cannot be completed uniquely.

In the Table 2.1, the critical sets contain quadruples (pairs of integers) only. Although all empty cells are known but we do not consider them as part of the critical set. If we consider that a set of quadruples along with all the empty cells form a critical set, then it is actually not a critical set but a uniquely completable set because some of the empty cells are redundant as shown in the example above. In the following section, we construct critical sets where empty cells are not known, i.e., nothing is known in advance except order of the Room square.

### 2.3 Critical sets when empty cells are unknown

Given the order of the Room square, a *critical set*  $Q = [Q_1, Q_2, Q_3, \dots, Q_c]$ ,  $|Q| = c$ , in a Room square  $R$  of order  $r$ , is a set of quadruples  $Q_a = [i, j; k \ l]$  and/or triples  $Q_a = [i, j; -]$  such that

- 1.  $R$  is the only Room square of order  $r$  which has the entry  $(k \ l)$  or  $(-)$  in the cell  $(i, j)$ , for each  $Q_a$ .
- 2. No proper subset of  $Q$  satisfies 1.

In  $Q_a$ ,  $(i, j)$  shows the position of the pair of integers  $(k \ l)$  or empty cell  $(-)$  in  $R$ .

**Example 2.5** A Room square of order 7 and one of its critical sets is given.

0 1	–	4 5	6 7	–	–	2 3
5 7	0 2	–	–	–	1 3	4 6
–	5 6	0 3	1 2	–	4 7	–
–	3 7	–	0 4	2 6	–	1 5
3 6	1 4	2 7	–	0 5	–	–
2 4	–	–	3 5	1 7	0 6	–
–	–	1 6	–	3 4	2 5	0 7

						4 6
			1 2			
	3 7					
	1 4	2 7				
			3 5		0 6	
				3 4	2 5	0 7

The critical set  $Q$  for the above Room square may be written as:  $Q = \{(2,7;4\ 6), (3,4;1\ 2), (4,2;3\ 7), (5,2;1\ 4), (5,3;2\ 7), (6,4;3\ 5), (6,6;0\ 6), (7,5;3\ 4), (7,6;2\ 5), (7,7;0\ 7)\}$ . This critical set contains  $10 \times 12 = 120$  bits of information.

Another critical set, which also contains 120 bits of information, is as follows:

		4 5	6 7			
	0 2				1 3	
	5 6		1 2			
		2 7				
				1 7		
				3 4		0 7

Completion Rules

We give five rules we use in completing a Room square. A Room square  $R$  of order  $r$  is written using  $r + 1$  elements and  $\frac{r-1}{2}$  empty cells in each row/column.

1. If any row  $i$  and column  $j$  have  $r$  elements already known, then if the  $(i, j)$ th cell is unknown, it is an empty cell.

2. If any row  $i$  and column  $j$  have  $r - 1$  elements and  $\frac{r-1}{2}$  empty cells already known, then if the  $(i, j)$ th cell is unknown, it is a pair of integers which have not already occurred.
3. If in row  $i$  and column  $j$ ,  $(r + 1 - k)$  elements and  $\frac{r-1}{2}$  empty cells are known and  $\binom{k}{2} - 1$  of the pairs, on the  $k$  unknown entries, already exist in the square, then if the  $(i, j)$ th cell is unknown, it is a pair of integers which have not already occurred.
4. If in row  $i$  and column  $j$ ,  $(r + 1 - k)$  elements are known and  $\binom{k}{2}$  of the pairs, on the  $k$  unknown entries, already exist in the square, then if the  $(i, j)$ th cell is unknown, it is an empty cell.
5. If after deleting an entry from a Room square, we are still able to uniquely complete that square, then that entry is not a member of the critical set.

Following are the algorithms to construct critical sets and search for minimal and maximal bounds of critical sets.

### 2.3.1 Algorithms to construct critical sets

Suppose  $R$  is the original Room square, take a Room square  $R'$  of order  $r$  corresponding to  $R$ . To find a critical set  $Q$ , take an empty Room square  $Q$  corresponding to  $R'$ . Select an entry at random from  $R'$ , delete it for the time being and try to complete  $R'$  using completion rules given above. If  $R'$  is completed without that entry, then delete that entry from  $R'$  permanently otherwise place it back into  $R'$  and move it to  $Q$  as well. Repeat this process of temporary deletion until none of the remaining entries in  $R'$  can be deleted. Hence  $Q$  is the *critical set* of  $R$ , since at this stage  $Q$  equals  $R'$  and no further entries can be deleted.

In the previous section, exhaustive search algorithms were developed to compute minimal and maximal critical sets when all empty cells are known. Since the number of inequivalent Room squares grows exponentially with the order, an exhaustive search to find minimal and maximal critical sets is not possible with computer resources available today. Here hill-climbing approach will be used to compute approximate bounds for minimal and maximal critical sets. The search space has been bounded by limiting the number of search attempts multiplied by the order of Room square.

*Small and large critical sets* are computed as:

Each time the program is executed for an instance of  $R$ , it takes a different seed value to generate a critical set of different size. For each instance of  $R$  of order 7, 9 and 11, the program has been executed  $10^8$  times but a lesser number of times for higher orders of  $R$ . The size of the critical set is recorded at each execution as a small or a large critical set. It is updated only if it is smaller than the previous critical set as *small cs* or larger than the previous one set as *large cs*.

*Upper bound for minimal critical set* is computed as:

1. Let  $Q^s$  be a small critical set of a Room square  $R$ , we search for a critical set consisting of  $|Q^s| - 1$  entries from  $R'$ .
2. For every partial Room square consisting of  $|Q^s| - 1$  entries taken at random from  $R$ :
  - (a) check whether it is a critical set or not.
    - i. If it is a critical set, then  $(Q^s - 1)$  is the new small critical set, replace the original  $Q^s$  set by the new set containing  $|Q^s| - 1$  entries, rename it as  $Q^s$  and now repeat the process from step 1.
    - ii. If it is not a critical set, then a minimal critical set for that square is the current  $Q^s$ .

*Lower bound for maximal critical set* is computed as:

Take a random set  $Q^t$  from  $R$  where  $t$  is the size of the large critical set.

1. check if it is uniquely completable, if it is, save it as  $Q^l$  ;
2. if it is not uniquely completable, systematically choose another element from  $R \setminus Q^t$  and form a new  $Q^t$  of size larger than previous one. check:
  - (a) if it is not uniquely completable, repeat step 2 until all choices are exhausted. If no candidate arises randomly, choose one of the sets at step 2 and repeat step 1.
  - (b) if it is uniquely completable, we have a new  $Q^l$  of larger size. Change  $t$  to  $t + 1$  and repeat the algorithm until no new  $Q^l$  arises.

To search for approximate bounds for minimal and maximal critical sets, at each stage, we allowed  $10^8$  attempts for  $R = 7, 9, 11$  and  $10^6$  attempts for  $R = 13, 15, 17$  and  $10^5$  attempts for  $R = 19, 21$  before we abandoned the search.

### 2.3.2 Cardinality

The algorithms for finding Room squares and their critical sets have been implemented in C++/Unix and were executed on SunSPARC workstations. Approximate bounds have been computed for minimal (*min. cs*) and maximal (*max. cs*) critical sets for all inequivalence classes of Room squares of order 7 [ $R_{11}$ ,  $R_{14}$ ,  $R_{15}$ ,  $R_{16}$ ,  $R_{44}$ ,  $R_{45}$ ] given by Wallis in [112]. The hill-climbing algorithm given by Dinitz and Stinson in [50] has been used to compute instances of Room squares of order 9 [ $R_{9_1}$ ,  $R_{9_2}$ ,  $R_{9_3}$ ,  $R_4$ ,  $R_{9_5}$ ,  $R_{9_6}$ ], order 11 [ $R_{11_1}$ ,  $R_{11_2}$ ,  $R_{11_3}$ ], order 13 [ $R_{13_1}$ ,  $R_{13_2}$ ,  $R_{13_3}$ ], order 15 [ $R_{15_1}$ ,  $R_{15_2}$ ,  $R_{15_3}$ ], order 17 [ $R_{17_1}$ ,  $R_{17_2}$ ,  $R_{17_3}$ ], order 19 [ $R_{19_1}$ ,  $R_{19_2}$ ] and order 21 [ $R_{21_1}$ ,  $R_{21_2}$ ].

Experimental results of the search are given in Table 2.2 below. At least one example of a critical set for each Room square of order less than or equal to 21 is given in Appendix A.2. Since an exhaustive search was not possible, there is no guarantee that the sizes of the critical sets given in the following table are minimal and maximal ones. In this table, size of the known *scs* of a Room square of order 7 is 10 and known *lcs* is 23:

Table 2.2: Bounds for *min. cs* and *max. cs*

Order of Room square	Class Representative	Cardinality	
		min. cs	max. cs
7	$R_{11}$	10	23
	$R_{14}$	11	22
	$R_{15}$	10	22
	$R_{16}$	10	22
	$R_{44}$	11	22
	$R_{45}$	10	23
9	$R_{9_1}$	19	38
	$R_{9_2}$	19	38
	$R_{9_3}$	19	39
	$R_{9_4}$	20	37
	$R_{9_5}$	19	38
	$R_{9_6}$	19	38
11	$R_{11_1}$	33	53
	$R_{11_2}$	33	54
	$R_{11_3}$	34	53
13	$R_{13_1}$	50	72
	$R_{13_2}$	51	74
	$R_{13_3}$	50	74
15	$R_{15_1}$	74	105
	$R_{15_2}$	74	105
	$R_{15_3}$	73	105
17	$R_{17_1}$	99	137
	$R_{17_2}$	99	133
	$R_{17_3}$	97	138
19	$R_{19_1}$	130	167
	$R_{19_2}$	133	163
21	$R_{21_1}$	165	208
	$R_{21_2}$	162	207

It is noted here that:  $scs(7) \leq 10 < \frac{r^2}{4}$  and  $scs(9) \leq 19 < \frac{r^2}{4}$

**Theorem 2.8** *If the same minimum information is required in addition to the rules to uniquely complete a partial Room square in all cases, then the cardinality of the*

*critical set, when empty cells are not part of the critical set, is always smaller than the cardinality of critical set when empty cells are included in the critical set.*

**Proof.** If a minimal critical set contains  $C_1(r)$  quadruples of integers then it contains  $4\lceil\log_2 r\rceil C_1(r)$  bits of information. If the minimal critical set contains  $C_2(r)$  quadruples of integers and  $C_3(r)$  empty cells then it contains  $4\lceil\log_2 r\rceil C_2(r) + (3\lceil\log_2 r\rceil + 1)C_3(r)$  bits of information. By assumption:

$$4\lceil\log_2 r\rceil C_1(r) = 4\lceil\log_2 r\rceil C_2(r) + (3\lceil\log_2 r\rceil + 1)C_3(r)$$

So if  $C_3(r) > 0$  then  $C_1(r) > C_2(r)$ . □

We considered critical sets in Room squares in order to obtain a comparison of their efficiency with those of latin squares. We noted that for a back-circulant latin square of order  $r$ , the critical set contains  $\lceil\frac{r^2}{4}\rceil \cdot 3\lceil\log_2 r\rceil$  bits of information. For  $r = 7$ , this gives 108 bits of information but back-circulant latin squares are heavily structured. For Room squares of order 7 we found:

- minimal critical sets of sizes 10 and 11 quadruples where no empty cells occur in the critical set. These use 120 and 132 bits respectively (see Table 2.2).
- We do not know but believe the inclusion of empty cells in the critical set will be less efficient. For order 7, this is  $12C_1(r) = 12C_2(r) + 7C_3(r)$  where experimentally we know  $C_1(r) = 10$  or  $11$ .

### 2.3.3 Complexity

Empirical results have led to the belief that running time  $T(n)$  of the algorithms to compute critical sets is  $\mathbf{O}(3^{\frac{n}{2}})$  where  $n$  is the order of Room square. The examples of critical sets given in this section and the previous section are strong critical sets, and therefore convenient to use, in the sense that the completion to a Room square is forced. But these are by no means the only critical sets. There may exist critical sets which cannot be completed by forcing. To consider this further, it is convenient to use the *defect graph*,  $G(P)$ , of a partial Room square  $P$ , defined as follows. The graph  $G(P)$  has one vertex for each incomplete row of  $P$ , one for each incomplete column, and one for each pair (entry) which is still available for placement in the completion of  $P$ . If the  $(i, j)$  cell of  $P$  is unknown, the  $G(P)$  contains the edge  $(r_i, c_j)$ ; if the pair  $(k, l)$  does not appear in row  $i$  of  $P$ , the graph  $G(P)$  contains the edge  $(r_i, e_{kl})$  or if the pair  $(k, l)$  does not appear in column  $j$  of  $P$ , then the graph  $G(P)$  contains the edge  $(c_j, e_{kl})$ .



Finding a completion of  $P$  is equivalent to finding a decomposition of the graph  $G(P)$  into edge-disjoint triangles. Using this observation, it can be shown that, in general, completing partial Room squares is NP-complete, even if both the partial square and its completion are required to be symmetric. Further, given a partial Room square and one completion, deciding whether a second completion exists is also NP-complete.

Colbourn, Colbourn and Stinson [36] make the observation that although the recognition of critical sets in latin squares in “special cases” where the unique completion of the partial latin square is straight forward this is not the case in general. They proved that “deciding whether a partial latin square has more than one completion is NP-complete, even if one completion is given as part of the problem description”.

Partial Room squares are much harder to complete because they have stricter conditions for completion than latin squares. So using the argument given in [36], it can be shown that, in general, completing partial Room squares is NP-complete. Further, given a partial Room square and one completion, deciding whether a second completion exists is also NP-complete. For general background in this area, see Garey and Johnson’s book [57]. An example of a partial Room square is given to show how hard it is to complete it:

Let us take second critical set of the Room square given in Example 2.5.

		4 5	6 7			
	0 2				1 3	
	5 6		1 2			
		2 7				
				1 7		
				3 4		0 7

If we remove the quadruple  $(1,3;4\ 5)$  from the critical set and try to complete the partial Room square, then there are 9 quadruples that can be tried in the cell  $(1,3)$ . These quadruples are:  $(1,3;0\ 1)$ ,  $(1,3;0\ 3)$ ,  $(1,3;0\ 4)$ ,  $(1,3;0\ 5)$ ,  $(1,3;1\ 4)$ ,  $(1,3;1\ 5)$ ,  $(1,3;3\ 5)$ ,  $(1,3;- \ -)$ . Result of each of these entries are given in Appendix A.4. Similarly each quadruple of the critical set was removed and all possible entries were tested at its place. Results are given in Appendix A.4 too. It is noted that none of the partial squares

completes to a Room square except one which is also inequivalent to original one and is given in Section 2.1.1. For higher orders of Room squares, it will be extremely difficult to complete a subset of a critical set.

## 2.4 Uniquely completable sets

In the previous two sections, the bounds for critical sets have been given on the basis of computational results. In this section, some general constructions of uniquely completable sets are given when empty cells are not known. A *uniquely completable set* is a partial Room square which completes uniquely to a Room square but may be able to have some entries removed to form a critical set.

**Example 2.6** A Room square of order 7, its uniquely completable set and its critical set are given respectively. The critical set given below was achieved by removing the entries (4 5) in the cell (1,3) and (1 5) in the cell (4,7) from the uniquely completable set.

0 1	–	4 5	6 7	–	–	2 3
5 7	0 2	–	–	–	1 3	4 6
–	5 6	0 3	1 2	–	4 7	–
–	3 7	–	0 4	2 6	–	1 5
3 6	1 4	2 7	–	0 5	–	–
2 4	–	–	3 5	1 7	0 6	–
–	–	1 6	–	3 4	2 5	0 7

		4 5				
						4 6
			1 2			
	3 7					1 5
	1 4	2 7				
			3 5		0 6	
				3 4	2 5	0 7

						4 6
			1 2			
	3 7					
	1 4	2 7				
			3 5		0 6	
				3 4	2 5	0 7

**Theorem 2.9** *When a row and a column of a Room square  $R$  of order  $r$  are missing, then  $R$  is uniquely completable.*

**Proof** Since one row and one column of  $R$  are missing, the entries in  $(r-1)$  rows and  $(r-1)$  columns of  $R$  are known. So there is only one unknown entry in each row and each column which is forced. Hence  $R$  is uniquely completable.

**Theorem 2.10** *When two rows (or two colmumns) of a Room square  $R$  of order  $r$  are missing, then  $R$  is uniquely completable up to permutation of pair chains.*

**Proof** We assume that two rows are missing. We recall a Room square of order  $r$  has  $r+1$  elements,  $\frac{1}{2}(r+1)$  pairs per row/column and  $\frac{1}{2}r(r+1)$  pairs in total. Each element must occur once in each of the extra rows. The partial Room square described above has  $\frac{1}{2}(r+1)(r-2)$  pairs completed. Thus the  $r+1$  pairs, not yet existing, can be listed. We list  $r+1$  pairs missing from the square and choose  $ab$  and  $ac$  to be two pairs in the list.  $ab, cd, ac, bd$  must take 4 columns whereas  $ab, cd, ef, cf, ae, bd$  take 3, 4, 5 or 6 columns. We call these pairs as “pair chain”. We note that pairs in a pair chain can be in the same column if all four elements in the two pairs are different. Hence we note each pair chain contains an even number of elements. We show the missing two rows can be completed using pair chains:

Step I: Once  $ab$  and  $ac$  have been chosen, we find the  $i$ th and  $j$ th columns of the partial Room square which do not contain  $a$  and fill the  $(r-1, i)$  cell with  $ab$  and the  $(r, j)$  cell with  $ac$ . Set  $x = b$ .

Step II: We now look in our list for the missing pair  $xy$ . There is now precisely one cell in one row and one column of the partial Room square containing no  $x$ . Fill this cell with  $xy$ . We now look in our list to find the element which has not yet occurred as pair with  $y$ . if  $yc$  is the pair missing, we fill the unique cell to complete the row

and column missing  $yc$  and then goto step III. If  $yz, z \neq c$ , is the pair missing , we set  $x = y$  and return to step II.

Step III: A pair chain has now completed. Each element in the pair chain now occurs an even number of times in the last two rows. If all pairs in our list are now used, then Room square is uniquely completable. Otherwise choose pairs  $de$  and  $df$  from our uncompleted list. Find the column which does not contain  $de$ , either one cell remains unfilled in the column which means this column can be filled uniquely or we have choice of rows. If  $de$  leads to a choice but  $df$  leads to a unique completion, set  $x = d$  and  $y = e$  or  $f$  according to which gives a unique completion and return to step II. The proof is similar when two columns are missing.

**Example 2.7** The following partial Room square of order 7, with two missing rows, is uniquely completable upto permutation of pair chains.

0 1	–	4 5	6 7	–	–	2 3
5 7	0 2	–	–	–	1 3	4 6
–	5 6	0 3	1 2	–	4 7	–
–	3 7	–	0 4	2 6	–	1 5
3 6	1 4	2 7	–	0 5	–	–

**Conjecture 2.11** When two rows and two columns of a Room square  $R$  of order  $r$  are missing, then  $R$  is uniquely completable up to permutation of pair chains.

**Example 2.8** The following partial Room square of order 7, with two rows and two columns missing, is uniquely completable upto permutation of pair chains.

0 1	–	4 5	6 7	–		
5 7	0 2	–	–	–		
–	5 6	0 3	1 2	–		
–	3 7	–	0 4	2 6		
3 6	1 4	2 7	–	0 5		

**Conjecture 2.12** When three rows or three columns of a Room square  $R$  of order  $r$  are missing, then  $R$  is uniquely completable up to permutation of pair chains.

**Example 2.9** The following partial Room square of order 7, with three columns missing, is uniquely completable.

0 1	4 5	2 7	–			
–	0 2	5 6	3 1			
–	–	0 3	6 7			
2 6	–	–	0 4			
–	3 7	–	–			
5 7	–	1 4	–			
3 4	1 6	–	2 5			

**Conjecture 2.13** *When two rows and three columns or three rows and two columns of a Room square  $R$  of order  $r$  are missing, then  $R$  is uniquely completable up to permutation of pair chains.*

**Theorem 2.14** *Let  $S$  be a Room square of order  $n$ . Let  $R$  be a subsquare of  $S$  of order  $r$ , and having critical set  $C$ . Then  $S$  has a critical set  $C^*$  contained in  $(S \setminus R) \cup C$ ,  $\iff (S \setminus R) \cup C$  is uniquely completable to  $S$ .*

**Proof  $\implies$ :** Suppose  $C^*$  is a critical set of  $S$ , with  $C^* \subseteq (S \setminus R) \cup C$ . Since  $C^*$  is a critical set of  $S$  then by definition,  $C^*$  uniquely completes to  $S$ . But any superset of  $C^*$  must also uniquely complete to  $S$ , for otherwise  $C^*$  will lie in at least two Room squares, which is a contradiction. That is,  $(S \setminus R) \cup C$  is uniquely completable to  $S$ .

**$\impliedby$ :** Suppose  $(S \setminus R) \cup C$  is uniquely completable to  $S$ . Let  $P = S \setminus R$ . If  $P \cup C$  is a critical set, then the Theorem is proved. Suppose not, then there exists an element  $p_1 \in P \cup C$  such that  $(P \setminus \{p_1\}) \cup C$  is uniquely completable to  $S$ . Since  $C$  is a critical set of  $R$ , therefore any non-trivial subset of  $C$  will be contained in at least two Room squares of order  $R$ , and hence it will also be contained in any two Room squares of order  $S$ . Thus  $p_1$  cannot be an element of  $C$ , and so  $p_1 \in P^*$ . Delete  $p_1$ . Let  $P^* = P \setminus p_1$ . If  $P^* \cup C$  is a critical set then we are done. Otherwise there exists an element  $p_2 \in P^*$ , such that  $P^* \setminus \{p_2\} \cup C$  is uniquely completable to  $S$ . Delete this element. Continuing as above, since  $S$  is finite we must eventually reduce to a critical set.

**Corollary 2.15** *Let  $S$  be a Room square of order  $n = 3r + 2$ . Let  $R$  be a subsquare of  $S$  of order  $r$ , and having critical set  $C$ . Then  $S$  has a critical set  $C^*$  contained in  $(S \setminus R) \cup C \iff (S \setminus R) \cup C$  is uniquely completable to  $S$ .*

**Corollary 2.16** *Let  $V$  be a Room square of order  $v = r(s - t) + t$ . Let  $R, S, T$  be subsquares of  $V$  of orders  $r, s, t$  respectively. Let  $R, S, T$  have critical sets of sizes  $R^*, S^*, T^*$  respectively. Then  $V$  has a critical set  $V^*$  contained in  $\{V \setminus (R \cup S \cup T)\} \cup (R^* \cup S^* \cup T^*) \iff \{V \setminus (R \cup S \cup T)\} \cup (R^* \cup S^* \cup T^*)$  is uniquely completable to  $V$ .*

## 2.5 Conclusion

- Our algorithms use exhaustive search techniques to look for minimal and maximal critical sets in Room squares. The algorithms are not efficient for larger Room squares. The algorithms need to be optimised or more efficient algorithms invented.
- Critical sets given in Section 2.3 are computed irrespective of the empty cells and are much better than critical sets given in Section 2.2. Critical sets, with empty cells known, are very expensive which is proved in Theorem 2.8 and some of the empty cells are even redundant.
- There is still not much known about critical sets in Room squares, so the bounds are rough and further research is needed to obtain theoretical bounds.
- Algorithm to find infinite families of critical sets as well as the smallest and largest ones need to be devised.

# Chapter 3

---

## Power, Influence and Strong Box

This chapter is based on the paper by Chaudhry and Seberry [28]. 95 percent of the work in this chapter is due to the first author.

### 3.1 Introduction

The structures which have rules for completion such as balanced incomplete block designs, latin squares, Room squares, F-squares, Youden squares, regular graphs, colourings, finite geometries and difference sets are considered. In particular the problem of unique completion of structures given the partial information is studied in this thesis. If the partial structure can be uniquely completed then this partial structure together with the rules contains the same information as the final structure. In this chapter, the information inherent in partial Room squares is studied, where it is not possible to uniquely complete the square. The influence and power of parts of the partial square on the unique completion of larger partial squares containing those parts are studied. That part of Room square, called the strong box, which is inaccessible to all the  $q - 1$  subsets of a critical set may be thought to contain the secret information. The size of the secret is also studied which will be used to model secret sharing schemes.

**Example 3.1** A critical set of a Room square  $R7_5$  of order 7 given by Wallis et al. in [112] and its completion.

0 1	* *	* *	* *	* *	* *	* *
* *	* *	* *	* *	* *	3 7	* *
2 7	5 6	* *	* *	* *	* *	* *
* *	* *	* *	* *	* *	* *	2 6
* *	4 7	* *	* *	0 5	* *	* *
* *	* *	* *	1 7	* *	* *	* *
* *	* *	* *	– –	2 3	4 5	* *

0 1	– –	– –	2 5	6 7	– –	3 4
4 6	0 2	– –	– –	– –	3 7	1 5
2 7	5 6	0 3	– –	1 4	– –	– –
– –	1 3	5 7	0 4	– –	– –	2 6
– –	4 7	– –	3 6	0 5	1 2	– –
3 5	– –	2 4	1 7	– –	0 6	– –
– –	– –	1 6	– –	2 3	4 5	0 7

where “\* \*” show the unknown entries and “–” show empty cells in the Room square.

### 3.2 Power, Influence and Strong Box

In a critical set  $\mathcal{Q}$  of size  $q$  of a Room square  $R$  of order  $r$ , every entry has a different importance in the reconstruction of  $R$ . If we delete an entry from  $\mathcal{Q}$  and try to reconstruct  $R$ , then the remaining  $(q - 1)$  entries will be able to recover some of the entries of  $R$ . The *power of an entry*,  $P$ , is the number of ways in which the Room square can be completed with that entry removed from  $\mathcal{Q}$ . The *power of a set*,  $\mathcal{P}$ , is the number of ways in which the Room square can be completed with that set of entries removed from  $\mathcal{Q}$ .

The *influence of an entry*,  $I$ , is the number of entries in the Room square which cannot be filled with that entry removed from  $\mathcal{Q}$ . The *influence of a set*,  $\mathcal{I}$ , is the number of entries in the Room square which cannot be filled with that set of entries removed from  $\mathcal{Q}$ . The deleted entry or set of entries from  $\mathcal{Q}$  which allows minimum number of fillings in  $R$  is called the *most influential* and is denoted by  $I_m$ . The deleted entry or set of entries from  $\mathcal{Q}$  which allows maximum number of fillings in  $R$  is called



the *least influential* and is denoted by  $I_l$ . A deleted entry or set of entries without which we cannot recover any entry in  $R$  has *perfect influence*. It is denoted by  $I_p$ .

The purpose of studying power and influence is to find the part of the Room square, *the strong box*, which cannot be uniquely completed by any set of  $q$ -subsets of the critical set of a Room square. The intersection of the cells of the influences of all the entries, one at a time, in the critical set is called the *strong box*. So no coalition of fewer than all the  $q$ -subsets of a critical set  $Q$  of size  $q$  can recover any entry from the strong box of Room square. Above definitions are illustrated with examples given below :

**Example 3.2** In the example 3.1, the size of the critical set is 11. In the following table, the influence of all eleven entries is given by deleting each entry, one at a time:

Deleted entry	Fillings	Influence	Remarks
1,1;0 1	10	38	$I_m \& I_p$
2,6;3 7	16	32	
3,1;2 7	14	34	
3,2;5 6	13	35	
4,7;2 6	14	34	
5,2;4 7	14	34	
5,5;0 5	18	30	$I_l$
6,4;1 7	16	32	
7,4; –	16	32	
7,5;2 3	13	35	
7,6;4 5	12	36	

In this table, the entry (1,1;0 1) has the most influence, it is also perfect as the remaining ten entries in  $Q$  do not permit the reconstruction of even a single entry of  $R$ . The entry (5,5;0 5) has the least influence because it reveals the most information about  $R$ . These most and least influence sets are shown in the squares given below respectively:

	* *	* *	* *	* *	* *	* *
* *	* *	* *	* *	* *		* *
		* *	* *	* *	* *	* *
* *	* *	* *	* *	* *	* *	
* *		* *	* *		* *	* *
* *	* *	* *		* *	* *	* *
* *	* *	* *				* *

		* *	* *	* *		* *
* *	* *	* *	* *	* *		* *
		* *	* *	* *		* *
* *	* *	* *	* *	* *		
* *		* *	* *		* *	* *
* *	* *	* *		* *	* *	* *

The strong box of the Room square in Example 1 consists of 29 unknown cells with “\* \*” as given below. These cells cannot be computed by any subset of the critical set:

		* *	* *	* *		* *
* *	* *	* *	* *	* *		* *
		* *	* *	* *		* *
* *	* *	* *	* *	* *		
* *			* *		* *	* *
* *	* *	* *		* *	* *	* *

After the permutation of rows and columns, the strong box has been shifted to the upper left corner of the square as shown below:

* *	* *	* *	* *	* *	* *	
* *	* *	* *	* *	* *	* *	
* *	* *	* *		* *	* *	* *
* *	* *	* *	* *			
* *	* *		* *			
		* *	* *	* *		* *

**Example 3.3** Take a critical set of size 10 of another Room square  $R7_1$  of order 7 given by Wallis et al. in [112] and compute the influence of all ten entries by deleting one entry at a time. In this example, an instance of perfect influence could not found by the deletion of just one entry, so perfect influences have been computed by deleting two entries at a time. The influences are given in the table below:

Deleted entry/set	Fillings	Influence	Remarks
1,4;2 5	24	24	$I_l$
1,5;4 6	11	37	$I_m$
2,3;1 5	12	36	
3,3;0 3	15	33	
4,5;2 7	16	32	
5,2;1 4	19	29	
5,7;2 3	19	29	
7,1;3 4	19	29	
7,6;1 2	18	30	
7,7;0 7	13	35	
1,5;4 6    7,7;0 7	8	39	$I_p$
2,3;1 5    7,7;0 7	8	39	$I_p$

In this example, most of the entries have differing influence. No single entry has a perfect influence. A set of two entries need both to be deleted to get perfect influence as shown in the table above. The strong box of the Room square, having this critical set, consists of 20 unknown cells.

Similarly the influences for other Room squares of order 7 have been computed and some examples of Room squares of order 9. In most examples of order 7, we need to delete two entries from the critical set to get perfect influence, but there might be other critical sets where one entry is sufficient to obtain perfect influence. We had to delete a set of three entries from a critical set of a Room square of order 9 to attain perfect influence. It has been found that entries in critical sets have different influence/importance, that is, some entries are more influential than others and vice-versa.

It is noticed that deletion of one entry from a critical set of a Room square of order 7 can give us perfect influence whereas in back-circulant latin squares, F-squares and

Youden squares, we have to delete two or more entries to get perfect influence. It is also noticed that sizes of the strong boxes in Room squares are much larger than those of latin squares, F-squares and Youden squares. The size of the strong box varies in Room squares from critical set to critical set, but roughly it is of size  $\frac{n^2}{2}$ . This makes Room squares more useful in applications as so little extra information can be gleaned. We have also observed that empty cells have less influence than pairs of integers in Room squares in the examples we studied.

### 3.3 Secret Sharing

A secret sharing scheme based on critical sets would have  $q$  sets of quadruples associated with the critical set distributed as shares. We allow the secret to be some function of the cells of quadruples in the strong box. If the secret is the same size as each share we obtain an ideal secret sharing scheme. By definition we have that no  $q - 1$  shares can recover the secret, that is, an outsider has the same probability of guessing the secret as an insider.

We now need to consider the possibility of guessing the contents of the strong box. The worst case is when  $q - 1$  cheaters have colluded to get as much information as they can before guessing the contents of the strong box. The power of the strong box, that is, the number of possible completions of the strong box to Room squares, is an NPC problem as proved by Colbourn, Colbourn and Stinson [34], that is “finding a completion of  $P$  is equivalent to finding a decomposition of  $G(P)$  into edge-disjoint triangles of the defect graph”. Using this observation, it has been shown that, in general, completing partial latin squares is NP-complete. Furthermore, given a partial latin square and one completion, deciding whether a second completion exists is also NP-complete. The same applies to Room squares as these are constructed from pairs of mutually orthogonal latin squares.

It is noted that most of the entries in the critical sets have differing influence, so a hierarchical scheme can be built as per the importance of the shares held by the share holders. In Example 2, the share holder having the share containing the quadruple (1,1;0 1) has perfect influence and can single handedly prevent reconstruction of any entry in the Room square whereas the share with (5,5;0 5) reveals eight entries of the

Room square.

## 3.4 Conclusion

The results in this chapter have been produced computationally. One of the open problems is to generalise these results and construct hierarchical structures of influences in Room squares and compute the power of the strong box if possible. These type of combinatorial structures can be used in secret sharing schemes, particularly hierarchical schemes where some participants are more important than others. It is noted that the size of the strong box in Room squares is much larger than those of latin squares, F-squares and Youden squares. Hence we conjecture that Room squares are more useful for applications where little information should be gleaned.

# Chapter 4

---

## Secret Sharing Schemes

Sections 4.2 and 4.3 are based on the paper by Chaudhry and Seberry [23]. More than 95 percent of the results are due to the first author. Section 4.4 is based on the paper by Chaudhry, Ghodosi and Seberry [26]. More than 60 percent of the work in this paper is due to the first author. Overall at least 85 percent of the results in this chapter have been contributed by the author of this thesis.

### 4.1 Introduction

In information based systems, the integrity of the information is commonly provided for by requiring that certain operation(s) can be carried out only by one or more participants who have access rights. Access is gained by a key, password or token, and governed by a secure key management scheme. If the key or password is shared between several participants in such a way that it can be reconstructed only by a responsible group acting in agreement, then a high degree of security is attained. Shared security systems of this sort are also used in financial institutions, in communication networks, in computing systems serving educational institutions and distribution environments. However, the best known examples of shared security systems are in the military: for instance, in activating a nuclear weapon, several senior officers must concur before the necessary password can be reconstructed. Another situation which motivates the subject of secret sharing is described:

*The head of an organisation keeps important documents in a safe of which only he or she knows the combination. However, the head is often absent for extended periods and occasionally information is needed from documents in the safe in order to maintain the day-to-day running of the organisation. The head deems it undesirable for the combination to be trusted to any one of the five executive board members. What is regarded as acceptable, however, is a compromise situation whereby at least two of the executive board members acting together can gain access to the safe.*

Can such a system be devised ?

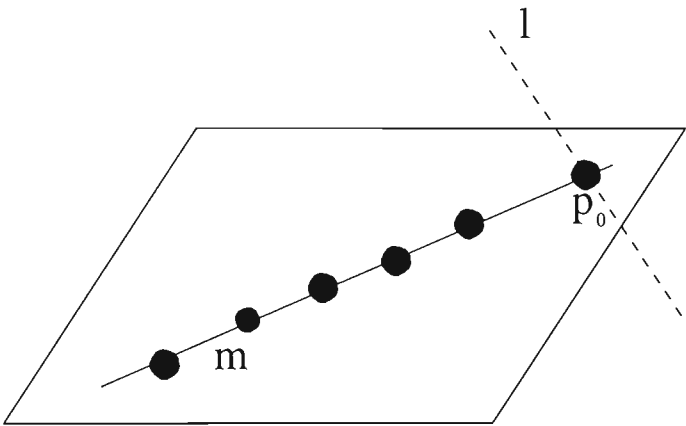


Figure 4.1: A projective space

Figure 4.1 provides a system to solve the above problem. The lines and points are chosen in projective space  $PG(2,q)$  where  $q \geq 5$ . It is publicly known that the safe combination is a point on line  $l$  but the actual point is kept secret. Each of the five executives is privately given a point on line  $m$  and the safe combination is chosen to be point  $p_0$ , the unique point of intersection of line  $m$  and line  $l$ . Any two executives can generate line  $m$  and hence evaluate  $p_0$  by intersecting  $m$  and  $l$ . However, one executive acting alone knows only that the safe combination must be one of the points on line  $l$  – the actual point remains secret since for every point  $p$  on line  $l$  there is a unique line passing through the executive’s point and  $p$ .

*Secret sharing schemes* are systems designed to solve problems of a similar type to the one we have just discussed. In general, there is a group of potential members of such a scheme and a collection of sets of these members which are desired to have access to some protected information. In the opening problem, the members are the five executives and the protected information is the combination to the safe.

This protected information might be of value itself or, as in the case of safe combination, might be of value in initiating some further action – in this case, the opening of the safe.

The information is protected by distributing to each member of the scheme an amount of partial information which relates in some way to the protected information.

This partial information is known only to the individual member to whom it is distributed and it is held secret by them. When any group of members of the scheme who are desired to have access to the protected information choose to do so, they can reconstruct it by pooling their pieces of partial information. Thus, in the opening problem, any two pieces of the partial information distributed to the executives must be sufficient to enable the combination of the safe to be determined.

Secret sharing schemes were first introduced by Blakley [18], Shamir [98] and Chaum [32] in 1979, and subsequently have been studied by numerous other authors. For a general discussion of shared secret schemes, see Simmons [100]. A number of mathematical structures have been used to model shared secret schemes. Some of these are polynomials, geometric configurations, block designs, Reed-Solomon codes, vector spaces, matroids, near-right fields, complete multipartite graphs, orthogonal arrays and latin squares. In this chapter, new secret sharing schemes based on critical sets of Room squares are proposed.

In most real-world applications there is also a need for a hierarchy to be built into the shared security system. That is, the key and password is shared between  $s$  individuals of rank  $1, \dots, r$  so that if a person of rank  $i$  is incapacitated, then a person of rank  $j \geq i$ , or a set of individuals of rank  $l < i$ , may replace the lost data. Brickell [19] and Simmons [100] have adapted the basic schemes and constructed multilevel schemes. Cooper, Donovan and Seberry [39] also proposed a multilevel scheme based on latin squares. A more formal terminology for secret sharing schemes is now introduced.

A *secret sharing scheme* is a method of sharing a secret  $S$  among a finite set of participants  $P$  in such a way that if the participants in  $A \subseteq P$  are authorised to know the secret, then by pooling together their shares, they can reconstruct the secret  $S$ ; but any set  $A \subset P$ , which is not authorised to know  $S$ , cannot reconstruct the secret. The key  $S$  is chosen by a special participant  $D$  called *dealer* and it is usually assumed that  $D \notin P$ . The dealer gives partial information called *share* to each participant to share the secret  $S$ .

An *access structure*  $\Gamma$  is the family of all the subsets of participants that are able to reconstruct the secret. An access structure is said to be *monotone* if any set which contains a subset that can recover the secret can itself recover the secret, i.e., if for



any subsets  $B$  and  $C$  of  $P$ , where  $B \subseteq C$  and  $B \in \Gamma$ , then  $C \in \Gamma$ . The subsets of  $P$  belonging to the access structure  $\Gamma$  are called *access sets* or *authorised sets* and those not belonging to the access structure are termed as *unauthorised sets*. The protected information is called the *secret* and the set  $K$  of all possible values of the secret is the *secret set*. A piece of partial information privately held by an individual participant is called a *share* and the set of all possible values of a share is known as the *share set*.

**Example 4.1** In Figure 4.1, the secret set are the points on line  $l$  and secret is point  $p_0$ . The shares are the five points on line  $m$  that are distributed to five participants. Finally, observing that  $P$  is the set of five points, the access structure is:

$$\Gamma = \{A \subseteq P : |A| \geq 2\}.$$

One property of a monotone access structure is that it has a unique collection of authorised sets of minimal size. Define  $B \in \Gamma$  to be a *minimal set* of  $\Gamma$  if  $C \in \Gamma, C \subseteq B$  implies  $C = B$ . The collection of all minimal sets of  $\Gamma$  is denoted by  $\Gamma^-$ .

**Example 4.2** Let  $P = \{A, B, C, D\}$  and  $\Gamma = \{ \{A, C\}, \{B, C\}, \{A, D\}, \{B, D\}, \{A, B, C\}, \{A, B, D\}, \{B, C, D\}, \{A, C, D\}, \{A, B, C, D\} \}$ .

Then  $\Gamma^- = \{ \{A, C\}, \{B, C\}, \{B, D\}, \{A, D\} \}$ .

It is often simpler to describe  $\Gamma$  in terms of its minimal set and thus the following are the three valid ways of representing  $\Gamma$  :

$$\Gamma = AC + BC + AD + BD, \Gamma = (A + B)C + (A + B)D \text{ and } \Gamma = (A + B)(C + D).$$

In general, if  $\Gamma^- = \{C_1, C_2, \dots, C_n\}$  where each  $C_i \subseteq P$ , then we represent this by  $\Gamma = C_1 + C_2 + \dots + C_n$ .

A  $(k, n)$  *threshold scheme* allows a secret to be shared among  $n$  participants in such a way that any  $k$  of them can recover the secret, but no group of  $k - 1$ , or fewer participants, can do so. A monotone access structure  $\Gamma$  defined on a participant set  $P$  such that  $|P| = n$  and  $\Gamma = \{X \subseteq P : |X| \geq k\}$  is known as  $(k, n)$  *threshold access structure*.

**Example 4.3** A  $(3, 3)$  threshold scheme

$$\text{Secret} = s, 0 \leq s \leq N - 1$$

Shares :  $x, y, z$  where  $0 \leq x, y \leq N - 1$

$$z = [s - (x + y)](mod N),$$

that is,  $x + y + z = s(mod N)$

Consider this example in a matrix form for  $N = 2$ :

s	x	y	z
0	0	0	0
0	0	1	1
0	1	1	0
0	1	0	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Each row represents a possible set of shares and their corresponding secret. Each row is also treated as rule. This matrix is made public. The dealer picks a rule at random and distributes shares to participants according to that rule. If all the three participants pool their shares, they can reveal the secret otherwise not.

**Example 4.4** A (2,3) threshold scheme

We take a latin square  $L$  of order 3.

1	2	3
2	3	1
3	1	2

Let  $S = (2, 1; 2), (3, 2; 1)(1, 3; 3)$  be the union of some critical sets of  $L$ . We can construct a 2-out-of-3 secret sharing scheme on this set. The latin square  $L$  is kept secret but its order is made public. When any two participants from  $S$  collaborate, they combine their shares and reconstruct the unique latin square containing their shares.

**Example 4.5** Shamir's  $(k, n)$  threshold scheme

Shamir's scheme [98] is based on interpolation of a polynomial defined over a finite field  $GF(q)$ . If  $k$  points are given in the two-dimensional plane,  $(x_i, y_i), i = 1, 2, \dots, k$ , there is a unique  $(k - 1)$ th degree polynomial,  $P^{k-1}(x)$ . If the secret key is taken to be an element  $p \in GF(q)$ , it can be partitioned into  $l$  shares as follows. A  $(k - 1)$ th degree polynomial  $P^{k-1}(x)$  is chosen randomly.  $P^{k-1}(x)$  has the representation

$$P^{k-1}(x) = \sum_{i=0}^{k-1} a_i x^i$$

where the constant term  $a_0$  is the secret  $p$ ; that is,  $P^{k-1}(0) = a_0 = p$ . The  $l$  shares,  $p_i = (x_i, y_i), l \geq k$ , are calculated by evaluating the above equation at  $l$  distinct points  $x_i, x_i \neq 0$ :

$$p_i = y_i = P^{k-1}(x_i)$$

In Figure 4.2, there is a polynomial of degree 2, knowledge of any three out of four or more points can recover the secret  $s$ . So it is 3-out-of-4 threshold scheme and also 3-out-of- $n$  scheme.

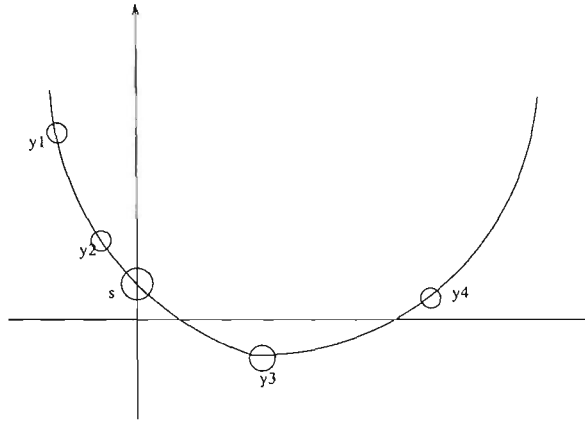


Figure 4.2: A polynomial of degree 2

A secret sharing scheme is said to be *perfect* provided the following two properties are satisfied:

- If an authorised subset of participants  $B \subseteq P$  pool their shares, then they can determine the value of  $S$ .
- If an unauthorised subset of participants  $B \subset P$  pool their shares, then they can determine nothing more than any outsider about the value of  $S$ .

The security of such a scheme is unconditional since no limit is placed on the amount of computation that can be performed by a subset of participants. If an unauthorised subset can obtain partial information regarding the secret, then the scheme is *non-perfect*.

**Example 4.6** In this example, a perfect secret sharing scheme is presented. Let  $\Gamma = ab + bc + cd$  be defined on participant set  $\{a, b, c, d\}$ . So the following matrix  $M$  is a  $PS(\Gamma, 2)$ :

$s$	$a$	$b$	$c$	$d$
0	0	0	0	0
0	0	0	1	1
0	1	1	2	0
0	1	1	3	1
1	0	1	0	1
1	0	1	1	0
1	1	0	2	1
1	1	0	3	0

To setup the scheme a row  $r$  must first be chosen at random. Participant  $p$  is then given  $M(r, p)$  and he must not reveal this value to any other participant or outsider to the scheme. The value of the secret is  $M(r, s)$ . When the time comes to reconstruct the secret, a group  $A \in \Gamma$  of participants present their values and scan the matrix  $M$ , then they can uniquely determine the secret. If  $A \notin \Gamma$ , they can determine nothing about the secret.

An *ideal* secret sharing scheme is a perfect scheme with the extra property that size of the share given to each participant is equal to the size of the secret.

**Example 4.7** The Shamir’s scheme mentioned in Example 4.5 is perfect as well as ideal. In that example, the polynomial is of degree two, so the knowledge of any three points can determine the polynomial but less than three points can determine nothing about the polynomial. So the scheme is perfect. Also size of the secret is one point on the polynomial and size of each share is one point too.

In a *multilevel scheme*, the participants are ranked and placed in levels  $r_1, \dots, r_w$ . Assume that there are  $l_i$  participants in level  $r_i$  for  $i = 1, \dots, w$ . So  $\sum l_i = l$ . A secret key  $S$  is chosen and  $l$  pieces of related information distributed, one piece to each participant. This is done in such a way that the secret can be recovered from the shares of  $t_i$  participants of rank  $r_i$ .

**Example 4.8** Consider the case of electronic transfer of funds between financial institutions. This transfer can only be initiated when an electronic signature is received. The signature will be reconstructed when the shares of two senior tellers and one vice president, or two vice presidents, is entered. The latin square in Example 4.4 is used

to construct a secret sharing scheme which satisfy these multilevel requirements. Let  $S = (2, 1; 2), (3, 2; 1), (1, 1; 1), (1, 2; 2)$  be the union of some critical sets of  $L$ . The access structure for this scheme will be based on the critical sets:

$$A_1 = (2, 1; 2), (1, 1; 1), (1, 2; 2)$$

$$A_2 = (3, 2; 1), (1, 1; 1), (1, 2; 2)$$

$$A_3 = (2, 1; 2), (3, 2; 1)$$

The latin square can be reconstructed from the shares  $(2, 1; 2)$  and  $(3, 2; 1)$ . However either of these two shares can be replaced by the two shares  $(1, 1; 1)$  and  $(1, 2; 2)$ . To satisfy the requirements of the model we could distribute the shares as follows:

- (1) The senior tellers each receive a share corresponding to each of the triples  $(1, 1; 1)$  and  $(1, 2; 2)$  respectively, and
- (2) the vice-presidents each receive a share corresponding to one of the triples  $(2, 1; 2)$  and  $(3, 2; 1)$ , respectively, Thus satisfying the requirement.

## 4.2 Secret sharing schemes using Room squares when all empty cells are known

A secret sharing scheme can be constructed in which the secret key is a Room square  $R$  of order  $r$ . This scheme exhibits the following characteristics: The Room square is taken to be the secret key and therefore kept private. However, the order of the Room square is made public. The shares in the secret are based on a partial Room square  $S = \{\cup A_i \mid A_i \text{ is a critical set in } R\}$ . The union can be taken over all possible critical sets in  $R$  or over some subset of critical sets. The number of critical sets used will be dependent on the order of the Room square and the number of participants in the secret sharing scheme. The access structure will be the set  $\Gamma = \{B \mid B \subseteq S \text{ and } B \supseteq A \text{ where } A \text{ is some critical set in } R\}$ . One can easily see that  $\Gamma$  is monotone. The protocol for secret sharing scheme, involving  $l$  participants and based on a Room square is as follows:

- A Room square  $R$  of order  $r$  is chosen. The number  $r$  is made public, but the Room square  $R$  is kept secret to be the key.
- A set  $S$  which is the union of a number of critical sets in  $R$  is defined.
- For each  $(i, j; k, l) \in S$ , the share  $(i, j; k, l)$  is distributed privately to a participant.

- When a group of participants whose shares constitute a critical set come together, they can reconstruct the Room square  $R$  and hence the secret key.

**Example 4.9** A Room square of order 7 and its critical sets

8,1	–	4,5	6,7	–	–	2,3
5,7	8,2	–	–	–	1,3	4,6
–	5,6	8,3	1,2	–	4,7	–
–	3,7	–	8,4	2,6	–	1,5
3,6	1,4	2,7	–	8,5	–	–
2,4	–	–	3,5	1,7	8,6	–
–	–	1,6	–	3,4	2,5	8,7

**	–	4,5	**	–	–	2,3
**	**	–	–	–	1,3	4,6
–	**	8,3	1,2	–	4,7	–
–	**	–	**	2,6	–	1,5
**	**	**	–	**	–	–
**	–	–	**	**	**	–
–	–	**	–	**	**	**

**	–	**	**	–	–	**
**	8,2	–	–	–	**	**
–	**	8,3	1,2	–	**	–
–	3,7	–	8,4	**	–	**
**	**	**	–	8,5	–	–
**	–	–	**	1,7	8,6	–
–	–	**	–	**	**	**

The first critical set can be written as:  $Q_1 = \{(1,3;4,5), (1,7;2,3), (2,6;1,3), (2,7;4,6), (3,3;8,3), (3,4;1,2), (3,6;4,7), (4,5;2,6), (4,7;1,5)\}$ . The second critical set can be written as:  $Q_2 = \{(2,2;8,2), (3,3;8,3), (3,4;1,2), (4,2;3,7), (4,4;8,4), (5,5;8,5), (6,5;1,7), (6,6;8,6)\}$ .

We demonstrate here how the scheme works on a small example and then give a more general construction.

Let  $S$  be the partial Room square  $\{(1,3;4,5), (1,7;2,3), (2,2;2,8), (2,6;1,3), (2,7;4,6), (3,3;8,3), (3,4;1,2), (3,6;4,7), (4,2;3,7), (4,4;4,8), (5,5;5,8), (4,5;2,6), (4,7;1,5), (6,5;1,7), (6,6;6,8)\}$ . All the parties are told that the order of the Room square is 7. Each participant is given a share  $(i, j; k, l)$ , for one such element of  $S$ . In order to recover the secret, an authorised group of participants must place their shares in a partial Room square. They then reconstruct the unique Room square containing these entries. These authorised groups are based on the critical sets contained in  $S$ . Two of the critical sets contained in  $S$  are:

$$A1 = \{(1,3;4,5), (1,7;2,3), (2,6;1,3), (2,7;4,6), (3,3;8,3), (3,4;1,2), (3,6;4,7), (4,5;2,6), (4,7;1,5)\}.$$

$$A2 = \{(2,2;2,8), (3,3;8,3), (3,4;1,2), (4,2;3,7), (4,4;4,8), (5,5;5,8), (6,5;1,7), (6,6;6,8)\}.$$

Note :  $|A1| = 9$  while  $|A2| = 8$ .

Now for a more general example, Let  $R$  be a Room square of order  $r$  and  $Q$  be a critical set. Define  $Q = \{Q' \mid Q' \text{ is the isotopic image of } Q\}$ . Let  $S' = \{Q' \mid Q' \in Q \text{ and } Q' \text{ is a critical set in } R\}$ . The protocol given above can be used to construct a secret sharing scheme where the shares are drawn from the set  $S'$ .

### 4.2.1 Key Management Scheme

Consider the situations where there are a number of secret sharing schemes all of which contain a common participant. This participant may be required to remember a number of shares. For example, a medical administrator (Registrar) may require access to several restricted files. These files may contain, say, patient data, hospital resources and organ bank data. Access to these files may be via a secret sharing scheme in which the registrar of the hospital always has a critical role. The registrar always has to remember several different shares. This obviously increases the complexity of the registrar's role and consequently reduces the security of the schemes.

In this section, a key management scheme is proposed in which a secret key is common to a number of secret sharing schemes. The shares related to this key are such that a primary share is held by one participant and this share is a necessary part of the reconstruction process in each scheme. Each scheme will involve a number of secondary shares which when combined with the primary share can be used to reconstruct the secret. It is also required that the secret can not be reconstructed uniquely from the combined information held by the secondary shares.

Inequivalent critical sets in a Room square can be used to model a key management scheme of this nature. This is illustrated with an example. Take the Room square of order 7 given in Example 4.9. Following are three distinct critical sets of this Room square which have the common pairs  $(3,3;8,3)$  and  $(3,4;1,2)$ .

$$A1 = \{(1,3;4,5), (1,7;2,3), (2,6;1,3), (2,7;4,6), (3,3;8,3), (3,4;1,2), (3,6;4,7), (4,5;2,6), (4,7;1,5)\}.$$

$$A2 = \{(2,2;2,8), (3,3;8,3), (3,4;1,2), (4,2;3,7), (4,4;4,8), (5,5;5,8), (6,5;1,7), (6,6;6,8)\}.$$

$$A3 = \{(1,1;1,8), (1,3;4,5), (1,4;6,7), (2,1;5,7), (2,7;4,6), (3,3;8,3), (3,4;1,2), (4,7;1,5), (1,5;3,6), (6,6;6,8)\}.$$

Note :  $|A1| = 9$ ,  $|A2| = 8$  and  $|A3| = 10$ .

Each department is assigned a different critical set  $A_i$  with the same participant receiving a share which is common to each  $A_i$ . In this case the registrar will be given the common share  $(3,3;8,3)$  or  $(3,4;1,2)$ . All departments will reconstruct the same secret, but each has a different set of keys to this secret. However, if all participants in the secondary (lower than registrar) level pool their shares, the secret cannot be reconstructed uniquely.

Another key management scheme can be developed to allow each department a different secret, but still have a common primary share using inequivalent Room squares of same order with some common pairs and same positions.

### 4.3 Generalised schemes using Room squares when empty cells are unknown

It is noted that *Threshold schemes* can only handle a fraction of secret sharing functions which one may wish to form. For example, if it is desirable to divide a secret among four participants  $A$ ,  $B$ ,  $C$  and  $D$  in such a way that either  $A$  together with  $B$  or  $C$  together with  $D$  can reconstruct the secret, then threshold schemes (even with weighting) are provably insufficient. To cope with such situations, Ito, Saito and Nishizeki [72] and Benaloh and Leichter [13] proposed methods of developing *generalised secret sharing schemes* to share a secret over an arbitrary access structure.

*Hierarchical schemes* are used in the situations where some shares are more important than others. Brickell [19], Simmons [100] and Cooper, Donovan and Seberry [39]



constructed hierarchical schemes. There are also scenarios where there are a number of secret sharing schemes all of which contain a common participant. These are called *key management schemes*. Motivation for the current section is to model new generalised secret sharing schemes using critical sets of Room squares.

**Example 4.10** A Room square of order 7 and one of its critical set of size 10 :

0,1	–	4,5	6,7	–	–	2,3
5,7	0,2	–	–	–	1,3	4,6
–	5,6	0,3	1,2	–	4,7	–
–	3,7	–	0,4	2,6	–	1,5
3,6	1,4	2,7	–	0,5	–	–
2,4	–	–	3,5	1,7	0,6	–
–	–	1,6	–	3,4	2,5	0,7

						4,6
			1,2			
	3,7					
	1,4	2,7				
			3,5		0,6	
				3,4	2,5	0,7

Hence a critical set  $\mathcal{Q}$  for the above Room square is :  $\mathcal{Q} = \{(2,7;4,6), (3,4;1,2), (4,2;3,7), (5,2;1,4), (5,3;2,7), (6,4;3,5), (6,6;0,6), (7,5;3,4), (7,6;2,5), (7,7;0,7)\}$ .

### 4.3.1 A Generalised Scheme

In a *general SSS*, the access structure is a collection of a set of authorised subsets of participants. In 1987 Ito, Saito and Nishizeki [72] described a general method of sharing a secret. Their method can be roughly described as : “For each of the (up to order  $2^{|P|}$ ) sets of the access structure  $P$ , divide the secret among each member of the set. Thus, in the worst case, each of the  $n$  participants may have to hold on the order of  $2^n$  shares”.

In 1988, Benaloh and Leichter [13] proposed a method of developing a generalised secret sharing scheme for any monotone access structure. Essentially, a key  $S$  is chosen

and for each set  $B$  of the access structure,  $|B|$  values  $s_1, \dots, s_{|B|}$  are selected such that the sum  $s_1 + \dots + s_{|B|} = S$ . They translated the access structure into a monotone formula. Each variable in the formula is associated with a participant in  $\mathcal{P}$ , and the value of the formula is true if and only if the set of variables which are true corresponds to a subset of  $\mathcal{P}$  which is in the access structure. They also proved that there exist monotone access structures for which there is no threshold scheme. But these schemes are not perfect and are also vulnerable to cheating.

We now model a generalised secret sharing scheme using critical sets of Room squares which prevents cheating and is computationally secure. Let a Room square  $R$  be taken to be the secret and its order  $r$  is made public. The shares in the secret are based on a partial Room square  $S = \{\cup Q_i \mid Q_i \text{ is a critical set in } R\}$ . The union of  $Q_i$  is taken over all possible critical sets in  $S$ . The number of critical sets used will be dependent on the order of the Room square and the number of participants in the secret sharing scheme. Let  $\mathcal{P} = \{P_1, \dots, P_n\}$  be the set of all participants in the scheme and let  $\Gamma = \{\mathcal{A}_1, \dots, \mathcal{A}_t\}$  be the general access structure with  $t$  authorised sets over  $\mathcal{P}$ . We consider  $Q_i$  is equivalent to  $\mathcal{A}_i$ . The  $i$ th authorised set  $Q_i = Q_{i1}, \dots, Q_{ij}$ ,  $1 \leq i \leq t$ , is the  $i$ th critical set of size  $j$  in  $R$ .  $Q_{ij}$  are the quadruples of  $i$ th critical set as mentioned in Section 4.2. In the protocol for generalised secret sharing, the dealer distributes the shares consisting of one or more quadruples  $Q_{ij} = (x_{ij}, y_{ij}; k_{ij}, l_{ij}) \in S$ , privately to each participant depending on the requirement of the scheme. When participants of one of the authorised sets  $\mathcal{A}_i$ , whose shares constitute a critical set, pool their shares together, they can reconstruct  $S$  and hence  $R$  which is the secret. An unauthorised group will not be able to reconstruct  $R$ .

**Example 4.11** It is shown now how the scheme works for Room square:

Take a Room square of order 7 given in Example 4.10. Let  $S$  be the partial Room square  $\{(2,1;5,7), (2,7;4,6), (3,4;1,2), (4,2;3,7), (5,1;3,6), (5,2;1,4), (5,3;2,7), (6,1;2,4), (6,4;3,5), (6,6;0,6), (7,5;3,4), (7,6;2,5), (7,7;0,7)\}$ .

The order of the Room square is 7 which is public. Each participant is allocated (privately) one or more quadruples  $(x, y; k, l) \in S$ . In order to recover the secret, an authorised group of participants must place their shares in a critical set. They then reconstruct the unique Room square containing this critical set. The authorised groups are based on the critical sets  $Q_1, Q_2$  and  $Q_3$  contained in  $S$  which are given as :

$$Q_1 = \{(2,7;4,6), (3,4;1,2), (4,2;3,7), (5,2;1,4), (5,3;2,7), (6,4;3,5), (6,6;0,6), (7,5;3,4), (7,6;2,5), (7,7;0,7)\}.$$

$$Q_2 = \{(2,7;4,6), (3,4;1,2), (4,2;3,7), (5,1;3,6), (5,2;1,4), (6,1;2,4), (6,4;3,5), (6,6;0,6), (7,5;3,4), (7,6;2,5), (7,7;0,7)\}.$$

$$Q_3 = \{(2,1;5,7), (2,7;4,6), (3,4;1,2), (4,2;3,7), (5,1;3,6), (5,2;1,4), (6,4;3,5), (6,6;0,6), (7,5;3,4), (7,6;2,5), (7,7;0,7)\}.$$

Now let  $\Gamma = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$  be an access structure over the set  $\mathcal{P} = \{P_1, \dots, P_5\}$  of five participants.  $\mathcal{A}_1 = \{P_1 \cup P_2\}$ ,  $\mathcal{A}_2 = \{P_1 \cup P_3 \cup P_4\}$  and  $\mathcal{A}_3 = \{P_1 \cup P_3 \cup P_5\}$ . The shares are distributed privately to the participants as :  $P_1 = \{(2,7;4,6), (3,4;1,2), (4,2;3,7), (6,6;0,6), (7,5;3,4), (7,6;2,5)\}$ ,  $P_2 = \{(5,2;1,4), (5,3;2,7), (6,4;3,5), (7,7;0,7)\}$ ,  $P_3 = \{(5,1;3,6), (5,2;1,4), (7,7;0,7)\}$ ,  $P_4 = \{(6,1;2,4), (6,4;3,5)\}$  and  $P_5 = \{(2,1;5,7), (6,4;3,5)\}$ .

When one of the three authorised sets in  $\Gamma$  collaborate, they will constitute one of the above critical sets and will be able to reconstruct the unique Room square which is the key, whereas an unauthorised set will not be able to reconstruct the secret.

### 4.3.2 Hierarchical Scheme

These schemes incorporate levels each of which has a number of authorised groups who can reconstruct the secret. The share of a participant at level  $i$  can be replaced by two or more participants at a lower level. For example, two members of the Joint Chiefs of Staff equal one President, it is easy to conceive of circumstances in which the President might wish to delegate authority to the Joint Chiefs to initiate some action with the provision that “if two of you agree that the circumstances warrant, then this is what you should do ....”. On the other hand, there are also plausible scenarios in which the concurrence of larger number of persons with lesser authority (and responsibility) could act in stead of smaller numbers of higher authority. For example, consider the case of an electronic transfer of funds between financial institutions. This transfer can only be initiated when an electronic signature is received. The signature will be reconstructed when the shares of two senior tellers and one vice-president or two vice presidents, are entered. Such schemes are described as hierarchical schemes.

In such schemes, The participants are ranked and placed in levels  $r_1, \dots, r_w$ . A secret key  $S$  is chosen and  $n$  pieces of related information distributed to each participant as per their rank. This is done in such a way that the secret can be recovered from the shares of specified participants of rank  $r_i$ . An incapacitated participant of rank  $r_i$  can

be replaced by a participant of rank  $r_j \geq r_i$ , or at least two participants of rank less than  $r_i$ .

Hierarchical schemes, based on critical sets of Room squares, can be developed in which some participants are more important than others. In hierarchical schemes, a participant at level  $j$  can be replaced by two or more participants at level  $i$ , where  $i < j$ . Let a Room square  $R$  is taken to be the secret and its order  $r$  is made public. The shares in the secret are based on a partial Room square  $S = \{\cup Q_i \mid Q_i \text{ is a critical set in } R\}$ . The union of  $Q_i$  is taken over all possible critical sets in  $S$ . Let  $\mathcal{P} = \{P_1, \dots, P_n\}$  be the set of all participants in the scheme and  $\mathcal{Q}_1 = \mathcal{Q}_{11}, \dots, \mathcal{Q}_{1j}$ , is the first access/critical set of size  $j$  in  $R$ .  $\mathcal{Q}_{1j}$  is the  $j$ th quadruple of first critical set as mentioned in Section 4.2. In the protocol for hierarchical scheme, the dealer distributes the shares consisting of two or more quadruples from  $\mathcal{Q}_i$  privately to each participant of rank  $r_1$ , one or more quadruples from  $\mathcal{Q}_i$  privately to each participant of rank  $r_2$  and quadruples of lesser importance to the participants of rank  $r_3$  from  $\mathcal{Q}_j$  and so on. When participants of  $\mathcal{Q}_i$ , whose shares constitute a critical set, pool their shares together, they can reconstruct the Room square  $R$  which is the secret. When a participant in  $\mathcal{Q}_i$  is absent, then two or more participants of rank  $r_2$  in  $\mathcal{Q}_i$  can replace them and find the secret. If a participant of rank  $r_2$  is missing then two or more participants of rank  $r_3$  from  $\mathcal{Q}_j$  can replace him. The hierarchy goes down as per the requirement of the scheme. This scheme is described in the example below.

**Example 4.12** We use a Room square of order 7 given in Example 4.10 to construct a hierarchical scheme which satisfy these requirements. Let an authorised set for this scheme be based on the critical set  $\mathcal{Q}_1 = \{(2,7;4,6), (3,4;1,2), (4,2;3,7), (5,2;1,4), (5,3;2,7), (6,4;3,5), (6,6;0,6), (7,5;3,4), (7,6;2,5), (7,7;0,7)\}$ . There are ten quadruples in this critical set. We distribute the shares to the participants privately to satisfy the requirements of the model as follows : Two vice presidents each receive a share, consisting of five distinct quadruples  $(x, y; k, l)$ , from the critical set. Ten senior tellers each receive a share, consisting of one distinct quadruple  $(x, y; k, l)$ , from the critical set. When two vice presidents pool their shares together, they can reconstruct the Room square which is the secret. Also one vice president and five senior tellers or all ten senior tellers can collaborate and reconstruct the secret. Even if one of the senior tellers is absent, two or more tellers can replace him. For instance, if the senior teller having the share  $(5,3;2,7)$  is absent then two tellers, each having one share  $(5,1;3,6)$  and  $(6,1;2,4)$  respectively, can replace him to reconstruct the secret as given in  $\mathcal{Q}_2$  in

Example 4.11.

### 4.3.3 Key Management Scheme

In this section, critical sets of Room squares are also used to construct a key management scheme in which a secret key is common to a number of secret sharing schemes. The shares related to this key are such that a primary share is held by one participant and this share is a necessary part of the reconstruction process in each scheme. Each scheme will involve a number of secondary shares which when combined with the primary share can be used to reconstruct the secret. Inequivalent critical sets in a Room square can be used to model a key management scheme of this nature.

Let a Room square  $R$  of order  $r$  is taken to be the secret and its order  $r$  is made public. The shares in the secret are based on a partial Room square  $S = \{\cup Q_i \mid Q_i \text{ is a critical set in } R\}$ . The union of  $Q_i$  is taken over all possible critical sets in  $S$ . Let  $\mathcal{P} = \{P_1, \dots, P_n\}$  be the set of all participants in the scheme and let  $\Gamma = \{\mathcal{A}_1, \dots, \mathcal{A}_t\}$  be the access structure with  $t$  authorised sets over  $\mathcal{P}$ . The  $i$ th authorised set  $\mathcal{A}_i = Q_i = Q_{i1}, \dots, Q_{ij}$ ,  $1 \leq i \leq t$ , is the  $i$ th critical set of size  $j$  in  $R$ . There is one primary share which is the integral part of each authorised set, whereas all other shares are secondary shares. In the protocol, the dealer distributes the shares, containing one or more quadruples  $Q_{ij} = (x_{ij}, y_{ij}; k_{ij}, l_{ij}) \in S$ , privately to each participant. When participants of every authorised set  $\mathcal{A}_i$  including primary share pool their shares, they can reconstruct the Room square. If all the participants holding secondary shares collaborate, they can't find the secret. We illustrate this with an example.

**Example 4.13** Consider three critical sets and the access structure given in example 4.11. In the access structure  $\Gamma = \{P_1 \cup P_2, P_1 \cup P_3 \cup P_4, P_1 \cup P_3 \cup P_5\}$ , the participant  $P_1$  has the primary share which is the necessary part of each authorised set. Each department is assigned a different critical set  $Q_i$ ,  $1 \leq i \leq 3$ , with the same participant  $P_1$  receiving a share which is common to each  $Q_i$ . In this case, the registrar will be given the quadruples which are in  $P_1$ . All three departments will reconstruct the same secret, but each has a different set of keys to this secret. However if all participants at the secondary  $(P_2, \dots, P_5)$  level pool their shares, the secret cannot be reconstructed uniquely.

### 4.3.4 Security of the schemes

The schemes proposed in this and previous sections have the following characteristics:

- Since the authorised groups are based on critical sets in Room squares, the absence of one share implies that secret cannot be recovered uniquely.
- In information theoretic sense, the schemes are obviously not perfect as an outsider can guess the secret from the set of all possible Room squares of order  $r$ . But the number of inequivalent Room squares of higher orders grows exponentially. So the probability of guessing a secret, consisting of a Room square of order greater than or equal to 11, is very small. Hence the schemes are computationally secure as proved by Colbourn, Colbourn and Stinson in [35].
- The structure of Room square is such that if the contents of one cell are changed, it will affect the whole array because Room square has strict conditions for its completion. If a participant tries to give a fake share, he will be caught immediately with very high probability. examples of fake shares are given in Appendix A, none of the fake shares could complete the square except one which was also different from the original square. So the schemes are secure against cheating as well.

## 4.4 A Perfect Scheme

In the previous sections, we proposed secret sharing schemes based on critical sets of Room squares when empty positions are known and unknown. These schemes are not perfect. In this section, we propose a perfect secret sharing scheme arising from critical sets of Room squares when empty positions are unknown. Though we propose secret sharing scheme based on Room squares, however, the method can easily be generalised to latin squares as well.

A secret sharing scheme is *perfect* if an unauthorised subset of participants  $\mathcal{B} \subset \mathcal{P}$  pool their shares, then they can determine nothing more than any outsider about the value of the secret  $S$ .

An authorised set  $\mathcal{A}$  is *minimal* if  $\mathcal{A}' \subset \mathcal{A}$  and  $\mathcal{A}' \in \Gamma$  implies that  $\mathcal{A}' = \mathcal{A}$ . We only consider *monotone* access structures in which  $\mathcal{A} \in \Gamma$  and  $\mathcal{A} \subset \mathcal{A}'$  implies  $\mathcal{A}' \in \Gamma$ . For such access structures, the collection of minimal authorised sets uniquely determines the access structure. In this section we use  $\Gamma$  to denote the representation of access structure in terms of minimal authorised sets.

Example 4.10 illustrates a Room square of order 7 and one of its critical sets of size 10. The critical set in the example consists of following quadruples:  $\{(2,7;4,6), (3,4;1,2), (4,2;3,7), (5,2;1,4), (5,3;2,7), (6,4;3,5), (6,6;0,6), (7,5;3,4), (7,6;2,5), (7,7;0,7)\}$ .

In the previous schemes, the shares of participants are the quadruples of a critical set taken from the Room square. When a group of participants, whose shares constitute a critical set, pool their shares together, they can reconstruct the Room square which is the key. But, every unauthorised set does not constitute the critical set, and thus, cannot reconstruct the secret. For example, in order to distribute the shares (the quadruples of the critical set given in Example 4.10) among an authorised set  $\mathcal{A}_i = \{P_{i_1}, P_{i_2}, P_{i_3}\}$ , the dealer may assign three quadruples  $(2,7;4,6)$ ,  $(3,4;1,2)$  and  $(4,2;3,7)$  to  $P_{i_1}$ , three quadruples  $(5,2;1,4)$ ,  $(5,3;2,7)$  and  $(6,4;3,5)$  to  $P_{i_2}$  and remaining four quadruples  $(6,6;0,6)$ ,  $(7,5;3,4)$ ,  $(7,6;2,5)$  and  $(7,7;0,7)$  to  $P_{i_3}$  (or any other possible combinations to distribute ten shares among three participants). A similar scheme was also proposed by Cooper et al [39] arising from latin squares. The shortcomings of the previous constructions are:

- i). The schemes are not perfect. Since each share is a component of a critical set, it determines the exact information of a component from the Room square and therefore, the uncertainty of a participant about the secret is not equal to the uncertainty of an outsider.
- ii). The scheme does not work if the number of participants in an authorised set is greater than the order of the critical set (since each participant must be assigned at least one quadruple).

Now we propose a perfect secret sharing scheme that is applicable over arbitrary access structures (no matter what is the size of its authorised sets). Though we propose secret sharing scheme based on Room squares, however, it can easily be generalised over latin squares as well.

#### 4.4.1 The Scheme

Let  $\mathcal{P} = \{P_1, \dots, P_n\}$  be the set of all participants in the system and let  $\Gamma = \{\mathcal{A}_1, \dots, \mathcal{A}_t\}$  be an access structure with  $t$  authorised sets over  $\mathcal{P}$ . Let the critical set  $\mathcal{Q} = \{Q_1, \dots, Q_c\}$  of a Room square  $R$  of order  $r$  be the secret<sup>1</sup>. For every authorised set  $\mathcal{A}_j$ ,  $1 \leq j \leq t$ ,

<sup>1</sup>In fact, the secret is the Room square  $R$ . However, from information point of view, the information contents of a Room square is the same as the information contents of its critical set.

of size  $n_j$ , the dealer uses the Karnin-Greene-Hellman [77] algorithm to distribute the shares to the participants.

#### Set-up Phase:

- i). For every participant  $P_{ju}$ ,  $1 \leq u \leq n_j - 1$ , the dealer,  $\mathcal{D}$ , selects (independently at random)  $c$  quadruples  $(x_{jv}, y_{jv}; k_{jv}, \ell_{jv})$ ,  $1 \leq v \leq c$ , from all possible values over  $(\mathbb{Z}_{r+1}, \mathbb{Z}_{r+1}, \mathbb{Z}_{r+1}, \mathbb{Z}_{r+1})$ .
- ii). The dealer computes the share for the last participant  $P_{jn_j}$ , corresponding to each  $Q_i = (x_i, y_i; k_i, \ell_i)$ ,  $1 \leq i \leq c$ , using

$$(x_{jn_i}, y_{jn_i}; k_{jn_i}, \ell_{jn_i}) = (x_i, y_i; k_i, \ell_i) - \left( \sum_{v=1}^{n_i-1} (x_{jv}, y_{jv}; k_{jv}, \ell_{jv}) \right) \quad (4.4.1)$$

where computation is done over  $\mathbb{Z}_{r+1}$ .

- iii).  $\mathcal{D}$  distributes, in private, the shares to the corresponding participants.

Clearly, if participants of an authorised set pool their shares (by adding their corresponding shares over  $\mathbb{Z}_{r+1}$ ) they can construct the critical set. Thus, the reconstruction phase could be as follows.

#### Secret Reconstruction Phase:

- i). Participants of every authorised set  $\mathcal{A}_i$  can pool their shares, that is, summation of all shares over  $\mathbb{Z}_{r+1}$  gives a critical set which is the secret.

**Example 4.14** Take a Room square of order 7 given in Example 4.10. Let the critical set  $\mathcal{Q} = \{(2,7;4,6), (3,4;1,2), (4,2;3,7), (5,2;1,4), (5,3;2,7), (6,4;3,5), (6,6;0,6), (7,5;3,4), (7,6;2,5), (7,7;0,7)\}$  be the secret,  $S$ .

Suppose there are three participants  $P_{11}$ ,  $P_{12}$  and  $P_{13}$  in the authorised set  $\mathcal{A}_1$ . Let the participants  $P_{11}$  and  $P_{12}$  be given the shares  $s_{11}$  and  $s_{12}$  (selected randomly) such that:

$$\begin{aligned} s_{11} &= \{(4,5;2,3), (3,4;5,5), (1,6;0,3), (2,3;1,5), (7,1;4,7), (4,4;0,7), (2,4;1,2), \\ &\quad (6,7;2,6), (0,0;3,5), (6,1;4,7)\}, \\ s_{12} &= \{(3,3;2,3), (4,7;1,0), (1,4;2,5), (5,7;6,7), (5,7;2,4), (3,7;3,4), (2,6;5,6), \\ &\quad (7,3;4,6), (7,5;0,4), (4,4;0,1)\}, \end{aligned}$$

The share  $s_{13}$  associated with participant  $P_{13}$  can be computed as follows (using equation (4.4.1) for every quadruple respectively),



$$\begin{aligned}
s_{13} &= S - (s_{11} + s_{12}) \\
&= \{(3,7;0,0), (4,1;3,5), (2,0;1,7), (6,0;2,0), (1,3;4,4), (7,1;0,2), (2,4;0,6), \\
&\quad (2,3;5,0), (0,1;7,4), (5,2;4,7)\}.
\end{aligned}$$

In secret reconstruction phase, when these three participants collaborate, (i.e., add their shares modulo 8) they can compute the critical set  $\mathcal{Q}$ , which is the secret.

#### 4.4.2 Security of the scheme

It is now proved that the proposed secret sharing scheme is perfect. That is, the uncertainty of a set of unauthorised collaborating participants (about the secret) is equal to the uncertainty of an outsider who knows nothing about the secret.

Let  $\mathcal{P} = \{P_1, \dots, P_n\}$  and let  $\Gamma = \{\mathcal{A}_1, \dots, \mathcal{A}_t\}$  be an access structure over  $\mathcal{P}$ . Let the critical set  $\mathcal{Q} = \{Q_1, \dots, Q_c\}$  of a Room square  $R$  of order  $r$  be the secret. Further, let a secret sharing scheme as mentioned earlier realises this access structure.

Observe that the  $n_j$  participants of every authorised set  $\mathcal{A}_j$  can recover the secret using equation (4.4.1). Now we have to show that any set  $\mathcal{B} \subset \mathcal{A}_j$  containing  $n_j - 1$  participants cannot recover the secret. Clearly, the first  $n_j - 1$  participants cannot do so, since they receive independent random tuples as their shares. Consider the  $n_j - 1$  participants in the set  $\mathcal{B}$  possess the shares  $s_{j_1}, \dots, s_{j_{i-1}}, s_{j_{i+1}}, \dots, s_{j_{n_j}}$  and the missing participant's share is  $s_{j_i}$  such that,

$$s_{j_i} = S - \sum_{\substack{u=1 \\ u \neq i}}^{n_j} s_{j_u} \pmod{r+1}.$$

By summing their shares, they can compute  $S - s_{j_i}$ . However, they do not know the random tuples of the share  $s_{j_i}$  and hence they have no information as to the real value of  $S$ . That is, the scheme is perfect.

### 4.5 Cheating prevention in secret sharing

In the previous sections, secret sharing schemes were proposed based on critical sets of Room squares. In the schemes, the shares of participants are the quadruples and/or triples of a critical set taken from the Room squares. When a group of participants, whose shares constitute a critical set, pool their shares together, they can reconstruct the Room square which is the key. But, every unauthorised set does not constitute the critical set, and thus, they cannot reconstruct the secret but may be able to obtain partial information depending on the size of the critical set. In a critical set, some shares

may be more powerful than others but none of them will be able to reconstruct the secret by collaborating with  $(n-2)$  participants. A similar scheme was also proposed by Cooper et al [39] arising from latin squares. These schemes do not take care of cheating detection and prevention. Shamir's scheme [98] was also protected from tampering by Wool and Tompa [109] but with certain assumptions. There are some other schemes on cheating detection but with certain restrictions. In this section, we model a secret sharing scheme in which cheating can not only be detected but also prevented without restrictions.

### 4.5.1 The Scheme

Let  $\mathcal{Q} = Q_1, Q_2, \dots, Q_c$ , is a critical set of a Room square  $R$  of order  $r$  where  $c$  is the size of the critical set. Let  $\mathcal{P} = \{P_1, \dots, P_c\}$  be the set of all participants in the scheme. The positions  $(x_i, y_i)$ ,  $1 \leq i \leq c$ , of the entries  $(k_i, l_i)$  of  $Q_i$  are made public and assigned to each participant  $P_i$ . The shares  $(k_i, l_i)$  are distributed to each participant  $P_i$  on a secure channel. The secret is the entry  $(k_0, l_0)$  which is found at public position  $(x_0, y_0)$ .  $(x_0, y_0; k_0, l_0)$  will be such quadruple which can only be found by the collaboration of all the participants in the scheme.

Clearly, if all the participants of the authorised set pool their corresponding shares they can construct the Room square and hence the secret  $(k_0, l_0)$ . The scheme is perfect as well as there is a very high probability of cheating prevention.

When any one of the participants tries to give a fake share, there are automatic ways which prevent him from doing so:

- i). A fake share may be one of the entries of a critical set, i.e., share of another participant.
- ii). An element of a fake share may already be present in the same row or column as part of a critical set.
- iii). Since each entry of a Room square has a unique position, a fake share might violate conditions of its completion. If the participant changes the contents of his share, the entry, the changed contents will have a different position in the square. If he changes the position of the entry, that position will have different contents in it.

**Example 4.15** Take a Room square of order 7 given in Example 4.10. Let  $\mathcal{Q} = \{(2,7;4,6), (3,4;1,2), (4,2;3,7), (5,2;1,4), (5,3;2,7), (6,4;3,5), (6,6;0,6), (7,5;3,4), (7,6;2,5),$

$(7,7;0,7)\}$  is one of the critical sets consisting of ten quadruples. First two digits (position) of each quadruple are public whereas last two digits (the pair) is the share. The positions  $\{(2,7), (3,4), (4,2), (5,2), (5,3), (6,4), (6,6), (7,5), (7,6), (7,7)\}$  are publicly known. The contents of respective portions  $\{(4,6), (1,2), (3,7), (1,4), (2,7), (3,5), (0,6), (3,4), (2,5), (0,7)\}$  are the shares privately given to each participant. There are ten participants  $\mathcal{P}_1, \mathcal{P}_2 \dots \mathcal{P}_{10}$  in the authorised set. Participant  $\mathcal{P}_1$  is given the share  $(4,6)$ ,  $\mathcal{P}_2$  is given the share  $(1,2)$  and so on.

In secret reconstruction phase, when these ten participants collaborate, (i.e., put their shares in the critical set) they can compute the Room square  $R$ , and hence the pair which is the secret.

## 4.6 Conclusion

In this chapter, secret sharing schemes based on critical sets of Room squares have been proposed. The schemes proposed in Section 4.3 are securer than those in Section 4.2 because critical sets with all empty cells known are much weaker than those without empty cells known. Empty cells leak a lot of information regarding the reconstruction of the Room square. It has also been shown by examples in Section 2.2 that some of the empty cells may even be redundant. The schemes proposed in Sections 4.2 and 4.3 are computationally secure as number of Room squares grows exponentially for higher order Room squares. Room squares are constructed under strict conditions and every entry in it has a particular location. So these schemes protect against cheating since a fake share will lead a participant to no completion at all or rarely more than one completions. A disadvantage of hierarchical scheme is that the more privileged members would be responsible for larger amounts of private information, and in the case of several levels perhaps responsible for too much information for them to feasibly handle securely but this is the requirement of these schemes.

Since there is little known about critical sets of Room squares, the implementation of these schemes is limited at the moment. However, the directions for future research are:

- Construction of families of critical sets for Room squares when all empty cells are unknown.
- Construction of families of minimal and maximal critical sets for Room squares.
- Quantify the security of the schemes more effectively, i.e., which critical sets are

more secure than others.

- A general process that will start with an access structure and result in a Room square.
- Most of the schemes rely on computational results; theoretical bounds need to be established for critical sets of Room squares.

## Part II

# Back-Circulant Latin Squares

# Chapter 5

---

## Uniquely Completable and Critical Sets

Sections 5.2 and 5.3 are based on the paper by Chaudhry, Seberry and Peddada [30]. The algorithm, some general constructions and most of the numerical results have been contributed by the first author. Sections 5.4 and 5.5 have been mainly done by Seberry but most of the numerical results are contributed by the author of this thesis. Overall about 45 percent of the work in this chapter has been done by the author of this thesis.

### 5.1 Introduction

Critical sets in latin squares have been studied by Nelder [87], Curran and van Rees [40], Smetaniuk [101], Stinson and van Rees [103], Cooper, Donovan and Seberry [37] and Gower [60]. Minimum defining sets of combinatorial designs, see for example [106], have been studied by Street, Sarvate, Kunkle, Seberry, Greenhill, Moran and Gower. This research has been motivated by studies of secret sharing schemes by Cooper, Donovan and Seberry [38], particularly they lend themselves easily to hierarchical and compartmentalised secret sharing, key distribution schemes by Merkle [84] and some problems in the design of experiments.

To find the precise minimum information needed to reconstruct a latin square, we need to have the minimum possible structure imposed on the latin square. However, for comparative purposes, it is valuable to compare the size of the least critical set under a number of scenarios.

A *latin square*  $L$  of order  $n$  is an  $n \times n$  array with elements chosen from a set  $N$ , of size  $n$ , that is  $1, 2, \dots, n$ , subject to the following condition :

- each of the elements  $1, 2, \dots, n$  occurs precisely once in each row of  $L$  and precisely once in each column of  $L$ ,

A *partial latin square*  $L$  of order  $n$  is an  $n \times n$  array with elements chosen from a set  $N$ , of size  $n$ , that is  $1, 2, \dots, n$ , subject to the following condition :

- each of the elements  $1, 2, \dots, n$  occurs at most once in each row of  $L$  and at most once in each column of  $L$ ,

A *uniquely completable set* is a partial latin square which completes uniquely to a latin square but may be able to have some elements removed to form a critical set. If a latin square  $L$  contains a  $s \times s$  subarray  $S$  and if  $S$  is a latin square of order  $s$ , then we say that  $S$  is a *latin subsquare* of  $L$ .

A *critical set* in a latin square  $L$  of order  $n$ , is a set  $A = \{(i, j; k) | i, j, k \in \{1, 2, \dots, n\}\}$  such that :

- $L$  is the only latin square of order  $n$  which has elements  $k$  in position  $(i, j)$  for each  $(i, j, k) \in A$ .
- no proper subset of  $A$  satisfies (i).

A *minimal critical set* in a latin square  $L$  of order  $n$  is a critical set of minimum cardinality.

In this chapter, we consider some even order latin squares of the form

$$\begin{bmatrix} A & B \\ B & C \end{bmatrix}$$

where  $A$  and  $B$  are back-circulant matrices and  $C$  has one of the three given types. We obtain new general constructions for two types of order  $n$ ,  $n$  is odd, which have uniquely completable sets of size  $\frac{13n^2+8n+3}{8}$  and  $\frac{13n^2-5}{8}$  respectively. We show that one of the subtypes, the flip back-circulant is equivalent to the general back-circulant matrix and another equivalence result. We also give new general constructions on uniquely completable sets in latin squares of orders  $2^t$  and  $mn$ .

## 5.2 Critical sets of small standard form latin squares

The size of the critical set for the following latin square of size 6 is 9 out of a possible 36 entries. We use the small squares on the upper and lower right-hand side of latin square to denote the size of critical set, (9), and total number of entries, (36), respectively.

In the following three arrays are a  $CS$ , the  $BC(3, 2)$  and  $BC(6)$  respectively.

1		3		5		9
3		5				
						2
5						
			2		4	36

1	2	3	4	5	6
2	1	4	3	6	5
3	4	5	6	2	1
4	3	6	5	1	2
5	6	2	1	4	3
6	5	1	2	3	4

1	3	5	2	4	6
3	5	2	4	6	1
5	2	4	6	1	3
2	4	6	1	3	5
4	6	1	3	5	2
6	1	3	5	2	4

We define as  $BC(n)$  or  $flipBC(n)$  as the latin square of order  $n$  which has subsquares of the form 

x	y
y	x

 above the back-diagonal and 

y	x
x	y

 below the back-diagonal.

We note that when the back-circulant latin square  $BC(n)$  is written using  $2 \times 2$  subsquares, the  $2 \times 2$  subsquares “flipover” when they re-occur at the right hand-side of the square. Observe  $2 \times 2$  squares

1	2
2	1

2	1
1	2

1	2	3	4	5	6	7	...	n-3	n-2	n-1	n
2	1	4	3	6	5	...	n-5	n-2	n-3	n	n-1
3	4	5	6	7	...	n-3	n-2	n-1	n	2	1
4	3	6	5	...	n-5	n-2	n-3	n	n-1	1	2
5	6	7	...	n-3	n-2	n-1	n	2	1	4	3
...											
n-3	n-2	n-1	n	2	1	4	3	6	...	n-6	n-5
n-2	n-3	n	n-1	1	2	3	4	...	n-6	n-5	n-4
n-1	n	2	1	4	3	6	...	n-4	n-5	n-2	n-3
n	n-1	1	2	3	4	...	n-6	n-5	n-4	n-3	n-2

$flipBC(m, 2)$  where  $m = \frac{n}{2}$

Curran and van Rees [40] and Smetaniuk [101] show that there is a uniquely completable set of the back-circulant latin square of order  $n$  of size  $\frac{n^2}{4}$ . Cooper, Donovan and Seberry [37] proved that it is a critical set. Using the following lemma, we establish that the critical set of the  $flipBC(m, 2)$  is also  $m^2 = \frac{n^2}{4}$ .

**Lemma 5.1** *Using the equivalence operations, interchange rows and columns. The  $BC(2m)$  is equivalent to the  $flipBC(m, 2)$ .*

**Proof.** Consider the  $BC(2m)$  given in Appendix B. Take the first,  $m + 1^{st}$ , then second,  $m + 2^{nd}$ , ...,  $m^{th}$ ,  $2m^{th}$  rows. Now reorder the columns taking the first,



$m + 1^{st}$ , second,  $m + 2^{nd}$ , ...,  $m^{th}$ ,  $2m^{th}$  columns. It is now easy to check that the matrix is composed of subsquares  $\begin{bmatrix} x & x + 1 \\ x + 1 & x \end{bmatrix}$  above the diagonal and subsquares  $\begin{bmatrix} y + 1 & y \\ y & y + 1 \end{bmatrix}$  below the diagonal, i.e.,  $flipBC(m, 2)$ . □

**Theorem 5.2** *The  $flipBC(m, 2)$  has a critical set of size  $\frac{n^2}{4}$ . It is :*

1		3		5		7		9	...	n-3		n-1		$\frac{1}{4}n^2$
								...						
3		5		7		9	...	n-3		n-1				
						...							2	
5		7		9	...	n-3		n-1						
				...								2		4
7		9	...	n-3		n-1								
⋮														
n-3		n-1								...				
					2		4		6	...	n-6		n-4	
n-1									...					
			2		4		6	...	n-6		n-4		n-2	$n^2$

**Proof.** This critical set is equivalent to that given in [40] by Lemma 5.1. □

### 5.3 Three constructions using subsquares

There are at least three essentially different constructions for an even order latin square from circulant subsquares: we use  $A$  or  $B$  for the circulant matrix,  $AR$  or  $BR$  for the back-circulant matrix and  $SA$  for the back-circulant matrix which has the first row of  $A$  reversed. Then the inequivalent reduced squares  $BC(2, n)$ ,  $BCFC(2, n)$  and  $MBC(2, n)$  have one of the two following forms:

$$\begin{bmatrix} AR & BR \\ BR & AR \end{bmatrix}$$

$$\begin{bmatrix} AR & BR \\ BR & A \end{bmatrix}$$

$$\begin{bmatrix} AR & BR \\ BR & SA \end{bmatrix}$$

So for order 6 we have  $BC(2, 3)$ ,  $BCFC(2, 3)$  and  $MBC(2, 3)$

1	2	3	4	5	6
2	3	1	5	6	4
3	1	2	6	4	5
4	5	6	1	2	3
5	6	4	2	3	1
6	4	5	3	1	2

1	2	3	4	5	6
2	3	1	5	6	4
3	1	2	6	4	5
4	5	6	1	2	3
5	6	4	3	1	2
6	4	5	2	3	1

1	2	3	4	5	6
2	3	1	5	6	4
3	1	2	6	4	5
4	5	6	3	2	1
5	6	4	2	1	3
6	4	5	1	3	2

It is easy to see these constructions are different. We call these the *construction using two back-circulants*  $BC(2, n)$ , *construction using back-circulants and forward circulants*,  $BCFC(2, n)$ , and *construction using a modified-two back-circulant*,  $MBC(2, n)$  respectively.

We write  $csBC(2, n)$ ,  $csBCFC(2, n)$  and  $csMBC(2, n)$  for critical sets of each of these types. Further examples of smaller  $csBC(2, n)$ ,  $csBCFC(2, n)$  and  $csMBC(2, n)$  are given in Appendix B.

We use a theorem of Cooper, Donovan and Gower [39] and denote it by *CDG*.

**Theorem 5.3 (CDG [39])** *There is a critical set of size  $\frac{13n^2-5}{8} = \frac{1}{2}(13m^2 + 13m + 2)$  in the  $BC(2, n) = BC(2, 2m + 1)$ .*

**Remark 5.3.1** *Cooper et al.'s small set has a large triangle with  $3m + 1$  entries in its base and four triangles with  $m$  entries in their bases, giving a total of*

$$\frac{1}{2}(3m + 1)(3m + 2) + 4 \times \frac{1}{2}m(m + 1) = \frac{1}{2}(13m^2 + 13m + 2)$$

*entries.*

### 5.3.1 Two Back-Circulant $[BC(2, n)]$

The square  $BC(s, t)$  is an  $s \times s$  back-circulant array containing back-circulant matrices of size  $t$ :

$$BC(2, n) = \begin{bmatrix} AR & BR \\ BR & AR \end{bmatrix}$$

$$BC(n, 2) = \begin{bmatrix} A_1 & A_2 & A_3 & \cdots & A_n \\ A_2 & A_3 & \cdots & A_n & A_1 \\ A_3 & \cdots & A_n & A_1 & A_2 \\ \vdots & & & & \\ A_{n-1} & A_n & A_1 & \cdots & A_{n-2} \\ A_n & A_1 & \cdots & A_{n-2} & A_{n-1} \end{bmatrix}$$

**Theorem 5.4** *Using the equivalence operations of interchange of rows and of columns, the  $BC(n, 2)$  is equivalent to the  $BC(2, n)$ .*

**Proof.** Consider the  $BC(n, 2)$  given above and reorder the rows by taking the first, third, fifth,  $\dots$ ,  $(n - 1)^{th}$ , second, fourth,  $\dots$ ,  $n^{th}$ . Now reorder the columns also taking the first, third, fifth,  $\dots$ ,  $(n - 1)^{th}$ , second, fourth,  $\dots$   $n^{th}$ .

The (1,1) element of each  $2 \times 2$  subsquare  $A_{ij}$  will now appear in the top left hand  $n \times n$  subsquare of the new matrix. As the  $A_{ij}$  were placed in a back-circulant arrangement this structure will be preserved in the  $n \times n$  subsquare. Similarly the (1,2) element of each  $2 \times 2$  subsquare  $A_{ij}$  will now appear in the top right hand  $n \times n$  subsquare of the new matrix. The (2,1) and (2,2) elements will give the bottom left and bottom right hand  $n \times n$  subsquares and have their back-circulant structure preserved.

The following example illustrates that the CDG Theorem does not give the smallest critical set as their theorem gives the size of  $csBC(2, 5)$  as 40 whereas we exhibit an example with size 32.

		3	4		6			9		32
	1				5	8				
			6			9				
	3			8		10			1	
						1				
	5	8		10				4	3	
7	8			1			4			
			9					6		
9			2		4		6	7		
			1			6				100

csBC(5,2)

### 5.3.2 Back-Circulant - Forward-Circulant [BCFC(2,n)]

In general this may be written as :

$$BCFC(2, n) = \begin{bmatrix} AR & BR \\ BR & A \end{bmatrix}$$

**Theorem 5.5** *There is a uniquely completable set of size  $\frac{13n^2+8n+3}{8} = \frac{1}{2}(13m^2+17m+6)$  in the  $BCFC(2, n) = BCFC(2, 2m + 1)$  of the form given in Appendix B.*

**Proof.** Let  $L$  be the back-circulant - forward-circulant (BCFC) latin square of order  $2n$  where  $n = 2m + 1$ . There are four subarrays in  $L$  each of order  $n$ . Let  $P$  be the partial latin square of  $L$  as given in Appendix B.4. We will show that at each step of the completion there exists a cell  $(i, j)$  and an element  $k$  such that  $k$  is the only entry which can be placed in this position. The following steps are used to show that  $P$  is uniquely completable to  $L$ .

1. The last  $2m+1$  columns can be filled uniquely in the first  $m + 1$  rows starting from top right-hand corner element. The last  $m$  columns are filled in top-right subarray.
2. The last  $2m+1$  rows can be filled uniquely in the first  $m + 1$  columns starting from bottom left-hand corner element. The last  $m$  rows are filled in bottom-left subarray.
3. The last  $m$  columns and last  $m$  rows can now be filled uniquely.
4. The  $(2m + 2)$ th row can be filled uniquely.
5. Now the bottom-left subarray can be filled uniquely except its upper right-hand part, also the bottom-right subarray can be filled uniquely except its upper left-hand part.
6. Finally we start completing the remaining  $(m + 2)$ th,  $(m + 3)$ th, ...,  $(2m + j)$ th rows/columns systematically.

At each step in the above argument there was an empty cell  $(i, j)$  which must contain a unique element  $k$ . Hence it follows that  $P$  has unique completion to  $L$ .  $\square$

A Critical Set for the MBC(2,7)

1	2	3	4	5	6	7	8	9	10					87
2	3	4	5	6	7	1	9	10						
3	4	5	6	7	1	2	10							
4	5	6	7	1	2	3								
5	6	7	1	2	3									11
6	7	1	2	3								11	12	
7	1	2	3									11	12	13
8	9	10								4	5	6	7	
9	10										4	5	6	
10												4	5	
							5							4
						11	4	5						
					11	12	3	4	5					
				11	12	13	2	3	4	5				196

5.3.3 Modified-two Back-Circulant [MBC(2,n)]

The modified-two back-circulant latin square has the first row of the right hand bottom subsquare reversed:

$$MBC(2,n) = \left[ \begin{array}{cc} AR & BR \\ BR & SA \end{array} \right] =$$

A	B	C	D	E	F
B	C	A	E	F	D
C	A	B	F	D	E
D	E	F	C	B	A
E	F	D	B	A	C
F	D	E	A	C	B

**Theorem 5.6** *There is a uniquely completable set of size  $\frac{13n^2-5}{8} = \frac{1}{2}(13m^2 + 13m + 2)$  for  $n < 9$  and  $\frac{14n^2-12n+30}{8}$  for  $n \geq 9$  in the  $MBC(2,n) = MBC(2,2m+1)$  case of the form given in Appendix B.*

**Proof.** Let  $L$  be the modified-two back-circulant latin square of order  $2n$  where  $n = 2m + 1$ . There are four subarrays in  $L$  each of order  $n$ . Let  $P$  be the partial latin square of  $L$  as given in Appendix B.4. We will show that at each step of the completion there exists a cell  $(i,j)$  and an element  $k$  such that  $k$  is the only entry which can be placed in this position. The following steps are used to show that  $P$  is uniquely completable to  $L$ .

1. The last  $2m + 1$  columns can be filled uniquely in the first  $m + 1$  rows starting from the top right-hand corner element. The last  $m$  columns are filled in the top-right subarray.
2. The last  $2m + 1$  rows can be filled uniquely in the first  $m + 1$  columns starting from the bottom left-hand corner element. The last  $m$  rows are filled in the bottom-left subarray.
3. The last  $m$  columns and last  $m$  rows can be filled uniquely.
4. The  $(2m + 2)$ th column and row can be filled uniquely.
5. The bottom-left subarray can be completed except for its upper right-hand part.
6. The top-right subarray can be completed except for its lower left-hand part.
7. The top-left subarray can be completed except for its lower right-hand part.
8. The bottom-right subarray can be completed except for its upper left-hand part.
9. Finally we start completing the remaining rows and columns systematically.

At each step in the above argument there was an empty cell  $(i, j)$  which must contain a unique element  $k$ . Hence it follows that  $P$  has unique completion to  $L$ .  $\square$

**Remark 5.3.2** *The uniquely completable set for  $MBC(2, n)$  ensures that  $AR$  contains  $1, \dots, n$  elements,  $BR$  contains  $n + 1, \dots, 2n$  elements and  $SA$  contains  $1, \dots, n$ .*

A Critical Set of MBC(2,5)										
1	2	3	4	5	6	7				40
2	3	4	5	1	7					
3	4	5	1	2						
4	5	1	2						8	
5	1	2						8	9	
6	7				5	4				
7					4					
				8					3	
			8	9				3	2	100

### 5.3.4 Algorithm

We now describe an algorithm to compute random critical sets which are of smaller size than those given by the theorems but which contain no pattern.

Step 1 Suppose  $L^0$  is the original latin square of order  $n$ . Take a latin square  $L^i$  corresponding to  $L^0$ . To find its critical set, take an empty latin square  $L^c$  corresponding to  $L^i$ . Select an element from  $L^i$ , delete it for the time being and try to complete  $L^i$  uniquely using rules for a latin square. There are three ways that an element can be forced: one can look at the row and column to find that which particular element has to go into the intersection of the row and column; one can look at a row and a particular element to see which column the element must go in that row, i.e., there are a total number of  $n - 1$  distinct columns in that row that are either filled or that column already contains that particular element; one uses a column and an element to find out that there is only one row in that column that could contain that element. If  $L^i$  is completed uniquely without that element, then delete that element from  $L^i$  permanently otherwise place it back into  $L^i$  and move it to  $L^c$  as well. Repeat this process of temporary deletion until none of the remaining elements in  $L^i$  can be deleted. Hence  $L^c$  is the critical set of that square, since at this stage it equals  $L^i$  and if we delete any further element from  $L^c$ , then the partial latin square can no longer be completed uniquely.

Since the number of inequivalent latin squares grows exponentially for higher orders of latin squares, an exhaustive search to find minimal critical sets is a hard problem. Here we use hill-climbing approach to compute approximate bounds for minimal critical sets. In our search, we have bounded the search by limiting the number of search attempts multiplied by the order of latin square. We now have a critical set  $L^c$  of size  $c$ .

Step 2 The program is executed for the original latin square  $L^0$ , by choosing a random starting cell to compute a random critical set. The size of the critical set found at each execution is noted. If it is smaller than the previously found smallest critical set, then the smallest set is updated. This way, we have discovered that we can decrease the size of critical set substantially, though we do not claim that this is the minimal one. This random selection of a starting element is repeated

on the order of  $10^8$  times for each instance of  $L^0$ .

The algorithm for finding critical sets has been implemented in C++/Unix and programs were executed on SunSPARC workstations. We have computed approximate bounds for minimal critical sets for small order latin squares which are better than those previously known, but with the patterned form, are given in the following table:

$n$	Total Elements	BC(2n)	$BCFC(2, n)$	$MBC(2, n)$
odd	$4n^2$	$\frac{13n^2-5}{8}$	$\frac{13n^2+8n+3}{8}$	$\frac{13n^2-5}{8}$
even	$4n^2$	$\frac{7n^2}{4}$	$\frac{13n^2+6n}{8}$	$\frac{13n^2-2n-16}{8}$
3	36	11(14)	11(18)	11(14)
4	64	22(28)	21(29)	21(23)
5	100	31(40)	36(46)	37(40)
6	144	57(63)	58(63)	(55)
7	196	76(79)	76(87)	(79)
8	256	(112)	99(110)	(100)
9	324	(131)	129(141)	(132)
10	400	(175)	170(170)	(158)

For odd  $n \geq 9$ , the  $UC$  set for  $MBC$  is  $\frac{14n^2-12n+30}{8}$ .

In this table, the number given in brackets is the upper bound on the size of uniquely completable or critical sets proved by the relevant theorem; the other number has been computed, using the above algorithm. For example 11(14) means that the theorem gave a construction of size 14 but there is a critical set of size 11 given in Appendix B.



## 5.4 Critical sets of latin squares of order $2^t$

In this section, we consider critical sets in the latin squares of order  $2^t$  which has the maximum number of subsquares i.e. if the rows/columns are labelled by the elements  $x_1, x_2, \dots, x_{2^t}$  of  $GF(2^t)$  then the contents of cell  $(x_1, x_2)$  is  $x_1 + x_2$ . If the  $x_1, x_2, \dots, x_{2^t}$  are in natural lexicographical order we say the square is reduced. We call this type  $B(2, 2, \dots, 2)$ .

We describe how the critical set of order 4 can be used to build a critical set for a latin square of order 8, which can be generalised to build a uniquely completable set for a latin square of order  $2^t$ , for  $t \geq 3$ . Critical sets of latin square of order 4 are:

1	2		
		4	
			2
	3		

		3	
2			
	4		2
			1

**Lemma 5.7** *There are 78 different critical sets of the latin square  $L_4=B(2,2)$  of the order 4, but only one if permutations of rows, columns and elements are considered.*

**Proof.** 78 different critical sets were found by undertaking a complete computer search of the  $\binom{16}{5}$  subsets of the  $B(2,2)$ . We now show that the critical set, described as follows, is unique up to permutations of rows, columns and elements :

- (i) there are 3 different elements, called say  $a, b, c$  which are in positions  $(i, j), (i, k)$  and  $(l, k)$ . We will call the element in the  $(j, k)$  element  $b$ , the pivotal element.
- (ii) there are exactly two other elements in the critical set, another use of the element  $b$  and a new element  $d, a \neq b \neq c \neq d$ . Now  $b$  and  $d$  must occur in either the squares  $(j, m)$  and  $(m, l)$  or  $(j, l)$  and  $(m, m)$  positions.

As columns  $m$  and  $l$  and rows  $j$  and  $m$  have not been labelled yet we can assume without loss of generality that the critical set is given by the diagram :

	j	k	m	l
i	a	b		
l		c		
j			b	
m				d

Consideration of row  $i$  and column  $l$  allows the  $(i, l) = c$ ,  $(i, m) = d$  to be decided. Consideration of row  $m$  and column  $k$  gives  $(m, k) = a$  and  $(m, j) = d$ . We now see  $(j, j) = c$ ,  $(j, l) = a$ ,  $(m, m) = c$ ,  $(l, m) = b$ ,  $(l, j) = d$ ,  $(l, l) = a$  and  $(m, j) = b$ . The structure of the  $B(2, 2)$  is now regained by choosing columns  $j, k, m, l$  to be column 1, 2, 4, 3 respectively and rows  $i, l, j, m$  to be 1, 4, 3, 2 respectively.  $\square$

Construction of the latin square  $L_8$  and its uniquely completable sets

The  $L_8 = B(2, 2, 2)$  is :

1	2	3	4	5	6	7	8
2	1	4	3	6	5	8	7
3	4	1	2	7	8	5	6
4	3	2	1	8	7	6	5
5	6	7	8	1	2	3	4
6	5	8	7	2	1	4	3
7	8	5	6	3	4	1	2
8	7	6	5	4	3	2	1

B(2,2,2)

Two uniquely completable sets for  $L_8$  are :

1	2	3	4			7		31
2	1	4	3	6				
3	4	1	2		8		6	
4	3	2	1				5	
		7				3		
6				2				
	8		6		4		2	
			5				1	64

1	2	3	4	5	6			31
2	1	4	3				7	
3	4	1	2		8			
4	3	2	1			6		
		7				3		
6				2				
	8		6		4		2	
			5				1	64

Note these partial latin squares illustrate that the uniquely completable sets of the latin square  $L_4 = B(2, 2)$  of order 4 are used to construct the latin square  $L_8$  of order 8. The latin square  $L_8$  of order 8 is basically a construction involving *four* latin squares of order 4.

**General construction of a latin square  $L_{2^t} = BC(2, ..., 2)$  using critical sets of the latin square  $L_{2^{t-1}} = BC(2, ..., 2)$**  Consider

(i) 
$$\begin{bmatrix} A & A_1 \\ A_2 & A_3 \end{bmatrix}$$

Where  $A$  is a complete latin square on the elements  $1, 2, \dots, n$  and  $A_1, A_2$  are critical sets on the elements  $n + 1, \dots, 2n$  and  $A_3$  is a critical set on the elements  $1, 2, \dots, n$ . Then (ii) gives a uniquely completable set in order  $2n$

(ii)

•	•	•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•	•	•
•	•	L	•	•	•	•	•	A <sub>1</sub>	•	•	•
•	•	•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•	•	•
•	•	A <sub>2</sub>	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	A <sub>3</sub>	•	•	•
•	•	•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•	•	•

and (iii) gives a uniquely completable set in order  $4n$

(iii)

$L_{2n}$				$L_n$		$C_n^1$	
				$C_n^2$		$C_n^3$	
$L_n$		$C_n^4$		$L_n$		$C_n^5$	
$C_n^6$		$C_n^7$		$C_n^8$		$C_n^9$	

**Theorem 5.8** *If  $A$  is a complete latin square on elements  $1, 2, \dots, n$  and  $A_1, A_2$  are critical sets on  $n + 1, \dots, 2n$  and  $A_3$  is a critical set on  $1, 2, \dots, n$  then (ii) is a partial latin square  $L_{2n}$  which is uniquely completable to a latin square.*

**Proof.** It is noted that having a complete latin square on elements  $1, 2, \dots, n$  means the  $n \times n$  righthand top and lefthand bottom squares must be filled by the elements  $n + 1, n + 2, \dots, 2n - 1, 2n$  only. As we have a critical set in these positions they can be uniquely completed to the subsquares on the required elements. The righthand bottom squares must be now filled with the elements  $1, 2, \dots, n$  and so, as it contains a critical set, can be uniquely completed.  $\square$

**Theorem 5.9** *Let  $|A|$  denote the number of elements in  $A$ . Now let*

$$B = \left[ \begin{array}{cc} A & A_1 \\ A_2 & A_3 \end{array} \right]$$

be a partial latin square of order  $2n$  where  $N(n)$  is the size of the smallest critical set for all  $B(2, 2, \dots, 2)$  of order  $n$ . Then,

$$\text{if } |A_1| = |A_2| = |A_3| = N(n) \text{ then } N(2n) \leq n^2 + 3N(n) \text{ and } N(4n) \leq 7n^2 + 9N(n).$$

### 5.5 Critical sets of latin squares of order $(mn)$

In this section, latin squares of order  $mn$  are considered. We describe how a uniquely completable set of order 4 can be used to build a uniquely completable set for latin square of order 12, which can be generalised to build a uniquely completable set for a latin square of order  $8t - 4$ , for  $t > 1$ . There are many inequivalent critical sets for the same latin squares. For example, both the following critical sets give the same  $B(2, 2)$ .

1	2	3	4
2	1	4	3
3	4	1	2
4	3	2	1

1	2		
		4	
			2
	3		

		3	
2			
	4		2
			1

#### Construction of a latin square $L_{12}$ and its uniquely completable set

1	2	3	4	5	6	7	8	9	10	11	12
2	1	4	3	6	5	8	7	10	9	12	11
3	4	1	2	7	8	5	6	11	12	9	10
4	3	2	1	8	7	6	5	12	11	10	9
5	6	7	8	9	10	11	12	1	2	3	4
6	5	8	7	10	9	12	11	2	1	4	3
7	8	5	6	11	12	9	10	3	4	1	2
8	7	6	5	12	11	10	9	4	3	2	1
9	10	11	12	1	2	3	4	5	6	7	8
10	9	12	11	2	1	4	3	6	5	8	7
11	12	9	10	3	4	1	2	7	8	5	6
12	11	10	9	4	3	2	1	8	7	6	5

$B(2,2,m)=BC(2,2,3)$

1	2	3	4	5	6	7	8	9	10			78
2	1	4	3	6	5	8	7			12		
3	4	1	2	7	8	5	6				10	
4	3	2	1	8	7	6	5		11			
5	6	7	8	9	10			1	2			
6	5	8	7			12				4		
7	8	5	6				10				2	
8	7	6	5		11				3			
9	10			1	2			5	6			
		12				4				8		
			10				2				6	
	11				3				7			144

#### Construction of a latin square $L_{20}$ and its uniquely completable set

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	1	4	3	6	5	8	7	10	9	12	11	14	13	16	15	18	17	20	19
3	4	1	2	7	8	5	6	11	12	9	10	15	16	13	14	19	20	17	18
4	3	2	1	8	7	6	5	12	11	10	9	16	15	14	13	20	19	18	17
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4
6	5	8	7	10	9	12	11	14	13	16	15	18	17	20	19	2	1	4	3
7	8	5	6	11	12	9	10	15	16	13	14	19	20	17	18	3	4	1	2
8	7	6	5	12	11	10	9	16	15	14	13	20	19	18	17	4	3	2	1
9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8
10	9	12	11	14	13	16	15	18	17	20	19	2	1	4	3	6	5	8	7
11	12	9	10	15	16	13	14	19	20	17	18	3	4	1	2	7	8	5	6
12	11	10	9	16	15	14	13	20	19	18	17	4	3	2	1	8	7	6	5
13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10	11	12
14	13	16	15	18	17	20	19	2	1	4	3	6	5	8	7	10	9	12	11
15	16	13	14	19	20	17	18	3	4	1	2	7	8	5	6	11	12	9	10
16	15	14	13	20	19	18	17	4	3	2	1	8	7	6	5	12	11	10	9
17	18	19	20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	17	20	19	2	1	4	3	6	5	8	7	10	9	12	11	14	13	16	15
19	20	17	18	3	4	1	2	7	8	5	6	11	12	9	10	15	16	13	14
20	19	18	17	4	3	2	1	8	7	6	5	12	11	10	9	16	15	14	13

B(2,2,m)=BC(2,2,5)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18			235
2	1	4	3	6	5	8	7	10	9	12	11	14	13	16	15			20		
3	4	1	2	7	8	5	6	11	12	9	10	15	16	13	14				18	
4	3	2	1	8	7	6	5	12	11	10	9	16	15	14	13		19			
5	6	7	8	9	10	11	12	13	14	15	16	17	18			1	2			
6	5	8	7	10	9	12	11	14	13	16	15			20				4		
7	8	5	6	11	12	9	10	15	16	13	14				18				2	
8	7	6	5	12	11	10	9	16	15	14	13		19				3			
9	10	11	12	13	14	15	16	17	18			1	2			5	6			
10	9	12	11	14	13	16	15			20				4				8		
11	12	9	10	15	16	13	14				18				2				6	
12	11	10	9	16	15	14	13		19				3				7			
13	14	15	16	17	18			1	2			5	6			9	10			
14	13	16	15			20				4				8				12		
15	16	13	14				18				2				6				10	
16	15	14	13		19				3				7				11			
17	18			1	2			5	6			9	10			13	14			
		20				4				8				12				16		
			18				2				6				10				14	
	19				3				7				11				15			400

General Construction

Suppose the squares are of the form :

$$\left[ \begin{array}{cccccc} A_1^* & A_2^* & \cdots & \cdots & A_{n-1}^* & C_n \\ A_2^* & A_3^* & \cdots & A_{n-2}^* & C_n & C_1 \\ A_3^* & \cdots & A_{n-1}^* & C_n & C_1 & C_2 \\ \vdots & & & & & \\ A_{n-1}^* & C_n & C_1 & \cdots & \cdots & C_{n-2} \\ C_n & C_1 & \cdots & \cdots & C_{n-2} & C_{n-1} \end{array} \right]$$

$BC(m,n)$

where  $A_i^*$  is a  $m \times m$  latin square with all its entries included and  $C_i$  is a critical set written on the elements. Then the number of elements in  $BC(m,n)$  is  $(i-1)n+1, (i-1)n+2, \dots, in$  is

$$m^2 \sum_{i=1}^{n-1} i + |c_i| \sum_{j=1}^n j.$$

where  $m^2$  is the number of elements in full square and  $|c_i|$  is the number of elements in  $C_i$ . Now

$\sum_{i=1}^{n-1} i$  is the number of full squares and  $\sum_{j=1}^n j$  is the number of partial squares so the total number of elements is

$$\equiv \frac{m^2(n-1)(n)}{2} + \frac{|c_i|n(n+1)}{2}.$$

**Theorem 5.10** *There is a uniquely completable set of size  $\frac{m^2(n-1)(n)}{2} + \frac{|c_i|n(n+1)}{2}$  for any  $n$  and  $BC(m, n)$  where  $|c_m|$  is the size of the appropriate critical set for latin square of size  $m$ .*

**Proof.** In  $BC(m, n)$ , it is observed that the elements in  $A_{n-1}^*$ , i.e.  $(m-2)n+1, (m-2)n+2, \dots, (m-1)n$ , all occur in every row and column except the last  $n$  rows and columns where they occur as a critical set  $C_{n-1}$ . Hence  $C_{n-1}$  can be completed uniquely to  $A_{n-1}^*$ . We now observe that the elements in both  $C_{n-2}'$ s have only one way to complete. Similarly we proceed through  $C_{n-3}$  to  $C_1$  and finally  $C_n$ . Thus we have uniquely completed the latin square.  $\square$

**Example 5.1** *If  $m = 4$  and  $L = B(2, 2)$  so that  $|c_4| = 5$ , we get*

$$\frac{1}{2} \times (16) \times n(n-1) + \frac{1}{2} \times 5n(n+1) = \frac{n(21n-11)}{2}.$$

# Chapter 6

---

## Nest, Influence and Latin Collection

This chapter is based on the paper by Fitina, Seberry and Chaudhry [56]. Most of the numerical results and initial general constructions have been contributed by the third author. Overall about 20 percent of the work in this chapter is due to the author of this thesis.

### 6.1 Introduction

In this chapter, the notions of nest and influence of a subset of a critical set of a back-circulant latin square are defined and their properties are studied. It is also shown that a secret sharing scheme based on a critical set of a latin square is both compartmentalised, and hierarchical. A *latin square*  $L$  of order  $n$  is a  $n \times n$  array with entries chosen from a set  $N$  of size  $n$ , such that each element of  $N$  occurs precisely once in each row and column. Thus  $L$  may be thought of as a set of triples  $(i, j; k)$ .  $L$  is said to be *back-circulant* if  $N = \{0, 1, \dots, n-1\}$ , and  $k = i + j \bmod n \forall i, j \in N$ . Let  $L$  be a back-circulant latin square of order  $n$  given by  $L = \{(i, j; i + j)\}$  with addition reduced modulo  $n$ . If  $A$  is any subset of  $L$ , we call the set of cells in  $A$  the *shape* of  $A$ .  $A$  is called a *critical set* of  $L$ , if (i)  $L$  is the only latin square of order  $n$  which has element  $k$  in position  $(i, j)$  for each  $(i, j; k) \in A$ , and (ii) no proper subset of  $A$  satisfies (i). A set  $A \subseteq L$  having property (i) is called *uniquely completable* and each element in it is also said to be uniquely completable. Every subset  $A$  of  $L$  is said to be *uniquely filled* or *fillable*. Let  $\frac{n-3}{2} \leq r \leq n-2$ . Donovan and Cooper [54] proved that the set

$$\begin{aligned} C = & \{(i, j; i + j) : i = 0, \dots, r \text{ and } j = 0, \dots, r - i\} \\ & \cup \\ & \{(i, j; i + j) : i = r + 2, \dots, n - 1 \text{ and } j = r + 1 - i, \dots, n - 1\} \end{aligned}$$



with  $\frac{n-3}{2} \leq r \leq n-2$ , is a critical set in  $L$ . Note that  $C$  is the union of two triangles, which we call the *upper triangle* and the *lower triangle*. If  $B \subseteq C$  is a subset of the critical set define the *nest*  $\mathcal{N}(B)$  of  $B$  to be the union of  $C \setminus B$  and the set that can be uniquely filled when  $B$  is deleted from  $C$ . Define the *influence-set* of  $B$ , denoted  $\mathcal{I}(B)$ , to be the shape of the set  $\{L \setminus \cup\{\mathcal{N}(b) : b \in B\}\}$ . The number  $|\mathcal{I}(B)|$  is called the *influence* of  $B$ , and denoted by  $\theta(B)$ . The set  $\mathcal{I}(C)$  is called the *strong box* of  $L$ . A set  $B \subset C$  is said to have *perfect influence* if  $B \cup \mathcal{N}(B) = C$ . A collection  $\mathcal{K}$  of partial latin squares  $I$  is called a *latin collection* if the entries in the cells of each row (and column) of each  $I \in \mathcal{K}$  are the same as those in the corresponding row (and column) of every other partial square in  $\mathcal{K}$ , and if the intersection of all the partial latin squares, regarded as sets of triples, is empty. A *latin interchange pair* is a latin collection of size 2. (See also [54] for an equivalent definition.) Elements of a latin interchange pair are called latin interchanges, and each is called the disjoint mate of the other. If  $A$  is a set and  $x$  is any element, we will often write  $xA$  for the set  $\{x\} \cup A$ . Let  $x, y \in C$ . We write  $x \rightsquigarrow y$  ( $x$  leads to  $y$ ) if there exist sets  $A, B \subseteq C$  with non-perfect influences, such that the following conditions hold:

LT1  $x \notin A, x \notin B, y \notin A, y \in B$

LT2  $xA$  has perfect influence

LT3  $A \cup B$  has perfect influence

LT4  $A \cup B'$  does not have perfect influence, for any proper subset  $B'$  of  $B$ .

We say  $x$  is *more influential than*  $y$ , and write  $x \longrightarrow y$  if

MI1 There is a finite sequence of the form:  $x \rightsquigarrow z_1 \rightsquigarrow z_2 \rightsquigarrow \dots z_m \rightsquigarrow y$

and

MI2 There is no finite sequence of the above form from  $y$  to  $x$ .

If  $x, y \in C$ , we define the relation  $x \sim y \iff \theta(x) = \theta(y)$ . Clearly  $\sim$  is an equivalence relation. For any  $x \in C$ , define  $[x]$  to be the equivalence class of  $x$ . Define the *index* of  $C$  to be the total number of influence classes.

Following example is given to illustrate some of these definitions:

**Example 6.1** Consider the critical set for the  $9 \times 9$  back-circulant latin square:

0	1	2	3					
1	2	3						
2	3							
3								
								4
							4	5
						4	5	6
					4	5	6	7

Let  $\alpha = (0, 0; 0)$ ,  $\beta = (8, 8; 7)$ ,  $\beta_1 = (8, 5; 4)$ ,  $\beta_{col} = (7, 8; 6)$ . The nests and influence-sets of these entries are as follows:

	1	2	3					
1	2	3						
2	3							1
3							1	2
						1	2	3
					1	2	3	4
				1	2	3	4	5
			1	2	3	4	5	6
		1	2	3	4	5	6	7

Nest of  $\alpha$

*				*	*	*	*	*
			*	*	*	*	*	*
		*	*	*	*	*	*	
	*	*	*	*	*	*	*	
*	*	*	*	*	*	*		
*	*	*	*	*	*			
*	*	*	*	*				
*	*	*	*					
*	*	*						

Influence-set of  $\alpha$

0	1	2	3	4	5	6		
1	2	3	4	5	6			
2	3	4	5	6				
3	4	5	6					
4	5	6						
5	6							4
6							4	5
						4	5	6
					4	5	6	

Nest of  $\beta$

							*	*
						*	*	*
					*	*	*	*
				*	*	*	*	*
			*	*	*	*	*	*
		*	*	*	*	*	*	
	*	*	*	*	*	*	*	
*	*	*	*	*	*	*		
*	*	*	*	*	*			*

Influence-set of  $\beta$

0	1	2	3			6	7	8
1	2	3				7	8	0
2	3					8	0	1
3						0	1	2
						1	2	3
						2	3	4
						3	4	5
						4	5	6
						5	6	7

Nest of  $\beta_1$

				*	*			
			*	*	*			
		*	*	*	*			
	*	*	*	*	*			
*	*	*	*	*	*			
*	*	*	*	*	*			
*	*	*	*	*	*			
*	*	*	*	*	*			
*	*	*	*	*	*			

Influence-set of  $\beta_1$

0	1	2	3	4	5			
1	2	3	4	5				
2	3	4	5					
3	4	5						
4	5							
5								4
							4	5
						4	5	
8	0	1	2	3	4	5	6	7

Nest of  $\beta_{col}$

						*	*	*
					*	*	*	*
				*	*	*	*	*
			*	*	*	*	*	*
		*	*	*	*	*	*	*
	*	*	*	*	*	*	*	
*	*	*	*	*	*	*	*	
*	*	*	*	*	*	*	*	*

Influence-set of  $\beta_{col}$

It can be seen on inspection, that the sets  $\{\alpha, \beta\}$  and  $\{\beta, \beta_1, \beta_{col}\}$  have perfect influence. Let  $A = \{\beta\}$ , and  $B = \{\beta_1, \beta_{col}\}$ . Clearly,  $\alpha \notin A$ ,  $\alpha \notin B$ ,  $\beta_1 \notin A$  and  $\beta_1 \in B$ .  $\alpha A = \{\alpha, \beta\}$  has perfect influence. Similarly,  $A \cup B = \{\beta, \beta_1, \beta_{col}\}$  has perfect influence. On the other hand, it can be easily verified that neither  $\beta_1 A = \{\beta, \beta_1\}$ , nor  $\beta_{col} A = \{\beta, \beta_{col}\}$  has perfect influence. Thus by definition,  $\alpha \rightsquigarrow \beta_1$ . Note also that  $\beta_1 \rightsquigarrow \alpha$ . Similarly  $\alpha \rightsquigarrow \beta_{col}$ , but it is not true that  $\beta_{col} \rightsquigarrow \alpha$ , therefore  $\alpha \longrightarrow \beta_{col}$ , or  $\alpha$  is more influential than  $\beta_{col}$ .

**Conjecture 6.1** For any  $x, y \in C$ ,  $x \rightsquigarrow y \implies \theta(x) \leq \theta(y)$ .

**Conjecture 6.2** Let  $L$  be a back-circulant latin square of odd order  $n$  and critical set  $C$ , with  $r = \frac{n-3}{2}$ . Then the index of  $C$  is  $r$ .

## 6.2 Some general results

In this section, the nests and the sizes of these nests for different entries of the critical set are calculated. These will be useful in the later sections where the influence of certain sets are calculated.

In the following lemmas, it is shown that some rows, columns and diagonals can be completed after the deletion of an element.

**Lemma 6.3** *Suppose  $(i, j; k)$  is deleted from the lower triangle of the critical set  $C$ . Then every column  $\lambda, j < \lambda \leq n-1$  can be uniquely filled, and every row  $\gamma, i < \gamma \leq n-1$  can similarly be uniquely filled.*

*Similarly if  $(i, j; k)$  was in the upper triangle then columns  $\lambda, 0 \leq \lambda < j$  can be filled, as can rows  $\gamma, 0 \leq \gamma < i$ .*

**Proof.** Neither row  $\gamma$  nor column  $\lambda$  depend on the information contained in row  $i$  or  $j$ , and thus can be filled uniquely.  $\square$

It is now shown that some diagonals can be filled.

**Lemma 6.4** *Let the critical set  $C$  be*

$$C = \{(i, j; i+j) : i = 0, \dots, r \text{ and } j = 0, \dots, r-i\}$$

$$\cup \{(i, j; i+j) : i = r+2, \dots, n-1 \text{ and } j = r+1-i, \dots, n-1\},$$

*where  $(n-3)/2 \leq r \leq n-2$ .*

1. *If  $(i, j; k)$  is deleted from the lower triangle, then every cell  $(u, v)$ , with  $0 \leq u, v < k$ , and  $r < u+v < k$  can be filled with entry  $u+v$ .*
2. *If  $(i, j; k)$  is deleted from the upper triangle, then every cell  $(u, v)$ , with  $k+1 < u, v \leq n-1$  and  $k < u+v \leq r$  can be filled with entry  $u+v$ .*

**Proof.** Only the first part will be proved, since the proof of the second part will be similar:

The set  $C$ , with  $(i, j; k)$  deleted from the lower triangle, and represented by  $*$ , is as follows:

0	1	...	r					
1	...	r						
...	r							
r								
								r + 1
								...
						r + 1	*	n - 3
					r + 1	...	n - 3	n - 2

The element  $r + 1$  occurs in rows  $r + 2$  to  $n - 1$ . Thus in column 0, there is only one cell that can be filled by  $r + 1$ , and that is position  $(r + 1, 0)$ . Fill this. Consider now cell  $(r, 1)$ . Element  $r + 1$  occurs once in each of the rows  $r + 1, \dots, n - 1$ . Thus there is only one cell that can be filled with  $r + 1$ , and that is  $(r, 1)$ . Similarly, one can show, that each of the other cells in the diagonal

$$\{(u, v; u + v) : 0 \leq u \leq r + 1, 0 \leq v \leq r + 1, u + v = r + 1\}$$

can be filled. A similar proof can be made for each of the other diagonals given above. □

**Lemma 6.5** *Suppose  $(i, j; k)$  is deleted from  $C$ . Apart from the empty positions given in Lemmas 6.1 and 6.2, no other empty position in the partial latin square can be uniquely filled.*

**Proof.** To help understand the proof we give below an example of a partial latin square of order 9. This partial latin square has been obtained by deleting the entry  $(7, 7; 5)$  from the critical set given in Example 6.1, and then obtaining the nest of this entry, by applying Lemmas 6.1 and 6.2. We claim that no empty entry in this partial latin square can be uniquely filled.

0	1	2	3	4				8
1	2	3	4					0
2	3	4						1
3	4							2
4								3
								4
							4	5
						4		6
8	0	1	2	3	4	5	6	7

We now give the proof of the lemma. Suppose that the entry  $(i, j; k = i + j)$  was deleted from the lower triangle in  $C$ . From Lemma 6.1 each of the rows  $\gamma, i < \gamma \leq n - 1$  can be uniquely filled, as can every column  $\lambda, j < \lambda \leq n - 1$ . By Lemma 6.2, each of the cells  $(u, v)$ , with  $0 \leq u, v < k$  and  $r < u + v < k$  can be uniquely filled.

Of the unfilled cells, the cells  $(r + 2, 0), (0, r + 2), (i, 0)$ , and  $(0, j)$  have the most information pertaining to them; that is, the union of the row and column containing each of these cells have more non-empty cells than any of the other cells.

Now row  $i$  contains each of the elements  $i, i + 1, \dots, k - 1, k + 1, \dots, r$ , and column  $j$  the elements  $j, j + 1, \dots, k - 1, k + 1, \dots, r$ . Thus any of the elements  $0, 1, \dots, \min\{i, j\} - 1, k, r + 1, r + 2, \dots, n - 1$  can fill cell  $(i, j)$ , and so  $(i, j)$  cannot be uniquely filled. Consequently no other empty cell in row  $i$  can be uniquely filled, and no other empty cell in column  $j$  can be uniquely filled. Thus neither cell  $(i, 0)$  nor cell  $(0, j)$  can be uniquely filled. Since cell  $(i, 0)$  cannot be uniquely filled, no empty cell in column 0 can be uniquely filled, and thus cell  $(r + 2, 0)$  cannot be uniquely filled. Similarly, since cell  $(0, j)$  cannot be uniquely filled, no empty cell in row 0 can be filled. Similar arguments can be used if an entry was deleted from the upper triangle in  $C$ . □

The above results are summarised as follows:

**Theorem 6.6** *Let  $C$  be the critical set given in the beginning. If an entry  $(i, j; k = i + j)$  is deleted from the lower triangle in  $C$ , then every column  $\lambda, j < \lambda \leq n - 1$ , and every row  $\gamma, i < \gamma \leq n - 1$  can be uniquely filled, as can every cell  $(u, v)$ , with  $r < u + v \leq k - 1$ . These are diagonals in the upper triangle, going from the upper right to the lower left. No other empty cell can be filled uniquely. Having filled these*

cells, the resulting partial latin square has size

$$|\mathcal{N}(\{(i, j; k)\})| = c_r - 1 + \sum_{\gamma=i+1}^{n-1} [n - (\gamma - (r + 1))] + \sum_{\lambda=j+1}^{n-1} [n - (\lambda - (r + 1))] + \sum_{\alpha=r+2}^k \alpha$$

where  $c_r$  is the size of the original critical set, the first summation is the number of entries filled in the rows  $\gamma$ , the second summation gives the number of entries filled in the columns  $\lambda$ , and the last summation gives the number of entries in the diagonals that are filled.

Similarly, if an entry  $(i, j; k)$  is deleted from the upper triangle in  $C$ , then

$$|\mathcal{N}(\{(i, j; k)\})| = c_r - 1 + \sum_{\gamma=0}^{i-1} [n + \gamma - r - 1] + \sum_{\lambda=0}^{j-1} [n + \lambda - r - 1] + \sum_{\alpha=n-r-1}^{n-k-2} \alpha$$

or

$$|\mathcal{N}(\{(i, j; k)\})| = c_r - 1 + \sum_{\gamma=1}^i [n + \gamma - r - 2] + \sum_{\lambda=1}^j [n + \lambda - r - 2] + \sum_{\alpha=n-r-1}^{n-k-2} \alpha$$

Here the first summation is the number of entries filled in the rows  $\gamma$ , the second summation is the number of columns filled, and the third summation is the number of diagonals that are filled.

**Proof.** The proof follows from the above lemmas. □

Now, the critical set has size:

$$c_r = \frac{1}{2}[(r + 1)(r + 2) + (n - r - 2)(n - r - 1)]$$

**Lemma 6.7** *The above summations may be simplified as follows:*

$$S1: \sum_{\gamma=i+1}^{n-1} [n - (\gamma - (r + 1))] = (n - i - 1)(n + r + 1) - \frac{1}{2}[n(n - 1) - i(i + 1)]$$

$$S2: \sum_{\lambda=j+1}^{n-1} [n - (\lambda - (r + 1))] = (n - j - 1)(n + r + 1) - \frac{1}{2}[n(n - 1) - j(j + 1)]$$

$$S3: \sum_{\gamma=1}^i [n + \gamma - r - 2] = \frac{i}{2}[2(n - r - 2) + (i + 1)]$$

$$S4: \sum_{\lambda=1}^j [n + \lambda - r - 2] = \frac{j}{2}[2(n - r - 2) + (j + 1)]$$

$$S5: \sum_{\alpha=r+2}^k \alpha = \frac{1}{2}[k(k + 1) - (r + 1)(r + 2)]$$

$$S6: \sum_{\alpha=n-r-1}^{n-k-2} \alpha = \frac{1}{2}[(n - k - 2)(n - k - 1) - (n - r - 2)(n - r - 1)]$$

It is noted that  $S1$  and  $S2$  are the same function, which we will denote by  $\underline{f}$ . This function denotes the total number of entries in the rows or columns that are filled, when an entry is deleted from the lower triangle of  $C$ . Also,  $S3$  and  $S4$  represent the same function, which is denoted by  $\bar{f}$ , which gives the total number of entries in the rows or columns that are filled, when an entry is deleted from the upper triangle of  $C$ . Relabel  $S5$  by the function notation  $\underline{g}$ , which gives the total number of entries in the diagonals that are filled when an entry is deleted from the lower triangle of  $C$ . Similarly, relabel  $S6$  by the function  $\bar{g}$ , which gives the total number of entries in the diagonals that are filled when an entry is deleted from the upper triangle of  $C$ . That is:

1.  $\underline{f}(r, x) = (n - x - 1)(n + r + 1) - \frac{1}{2}[n(n - 1) - x(x + 1)]$
2.  $\bar{f}(r, x) = \frac{x}{2}[2(n - r - 2) + (x + 1)]$
3.  $\underline{g}(k, r) = \frac{1}{2}[k(k + 1) - (r + 1)(r + 2)]$
4.  $\bar{g}(k, r) = \frac{1}{2}[(n - k - 2)(n - k - 1) - (n - r - 2)(n - r - 1)]$

Then,

**Theorem 6.8** *The number of cells that can be completed after deletion of an element from the lower triangle and upper triangle, respectively, are:*

(i) *If  $(i, j; k)$  is deleted from the lower triangle in  $C$ , then*

$$|\mathcal{N}((i, j; k))| = \underline{f}(r, j) + \underline{f}(r, i) + \underline{g}(k, r) + c_r - 1$$

and

(ii) *if  $(i, j; k)$  is deleted from the upper triangle in  $C$ , then*

$$|\mathcal{N}((i, j; k))| = \bar{f}(r, j) + \bar{f}(r, i) + \bar{g}(k, r) + c_r - 1$$

**Theorem 6.9** *If  $x$  and  $y$  are any two triples in  $C$ , then  $\mathcal{N}(\{x, y\}) = \mathcal{N}(\{x\}) \cap \mathcal{N}(\{y\})$ .*

**Proof.** Since  $\{x, y\} \supset \{x\}$  therefore  $\mathcal{N}(\{x, y\}) \subseteq \mathcal{N}(\{x\})$ . Similarly,  $\mathcal{N}(\{x, y\}) \subseteq \mathcal{N}(\{y\})$ . Thus taking intersections, we get  $\mathcal{N}(\{x, y\}) \subseteq \mathcal{N}(\{x\}) \cap \mathcal{N}(\{y\})$ .

We now need to show that  $\mathcal{N}(\{x\}) \cap \mathcal{N}(\{y\}) \subseteq \mathcal{N}(\{x, y\})$ . If  $z \in \mathcal{N}(\{x\}) \cap \mathcal{N}(\{y\})$ , then  $z \in \mathcal{N}(\{x\})$  and  $z \in \mathcal{N}(\{y\})$ . That is,  $z$  can be uniquely filled after both elements  $x$  and  $y$  have been deleted from  $C$ . But then  $z \in \mathcal{N}(\{x, y\})$ .  $\square$



**Theorem 6.10** *If  $b_1, b_2, \dots, b_n$  are any  $n$  distinct elements in  $B$ , then*

$$\mathcal{N}(\{b_1\}) \cap \mathcal{N}(\{b_2\}) \cap \dots \cap \mathcal{N}(\{b_n\}) = \mathcal{N}(\{b_1, b_2, \dots, b_n\}).$$

**Proof.** Trivially,

$$\mathcal{N}(\{b_1, b_2, \dots, b_n\}) \subseteq \mathcal{N}(\{b_1\}) \cap \mathcal{N}(\{b_2\}) \cap \dots \cap \mathcal{N}(\{b_n\}),$$

so we need to show that

$$\mathcal{N}(\{b_1\}) \cap \mathcal{N}(\{b_2\}) \cap \dots \cap \mathcal{N}(\{b_n\}) \subseteq \mathcal{N}(\{b_1, b_2, \dots, b_n\}).$$

Suppose that for some  $k \leq n$ ,

$$\mathcal{N}(\{b_1\}) \cap \mathcal{N}(\{b_2\}) \cap \dots \cap \mathcal{N}(\{b_k\}) \subseteq \mathcal{N}(\{b_1, b_2, \dots, b_k\}).$$

We want to show that

$$\mathcal{N}(\{b_1\}) \cap \mathcal{N}(\{b_2\}) \cap \dots \cap \mathcal{N}(\{b_{k+1}\}) \subseteq \mathcal{N}(\{b_1, b_2, \dots, b_{k+1}\}).$$

Now

$$\begin{aligned} & \mathcal{N}(\{b_1\}) \cap \mathcal{N}(\{b_2\}) \cap \dots \cap \mathcal{N}(\{b_k\}) \cap \mathcal{N}(\{b_{k+1}\}) \\ & \subseteq \mathcal{N}(\{b_1, b_2, \dots, b_k\}) \cap \mathcal{N}(\{b_{k+1}\}) \end{aligned}$$

If  $z \in \mathcal{N}(\{b_1, b_2, \dots, b_k\}) \cap \mathcal{N}(\{b_{k+1}\})$  then  $z \in \mathcal{N}(\{b_1, b_2, \dots, b_k\})$  and  $z \in \mathcal{N}(\{b_{k+1}\})$ . That is,  $z$  can be uniquely filled after the elements  $b_1, b_2, \dots, b_k, b_{k+1}$  are deleted from  $C$ , and thus  $z \in \mathcal{N}(\{b_1, b_2, \dots, b_{k+1}\})$ . From this and the above theorem the proof follows.  $\square$

**Corollary 6.11** *Let  $L$  and  $C$  be as above.*

1. *Then for any set  $B \subseteq C$ ,*

$$\mathcal{N}(B) = \cap \{\mathcal{N}(\{b\}) : b \in B\};$$

*and*

2. *For any two sets  $A, B \subseteq C$ ,  $\mathcal{N}(A \cup B) = \mathcal{N}(A) \cap \mathcal{N}(B)$ .*

**Proof.** We shall only prove 2. Let  $A = \{a_1, \dots, a_s\}$ , and  $B = \{b_1, \dots, b_t\}$ . Then  $\mathcal{N}(A \cup B) = \mathcal{N}(\{a_1, \dots, a_s, b_1, \dots, b_t\}) = \mathcal{N}(\{a_1\}) \cap \dots \cap \mathcal{N}(\{a_s\}) \cap \mathcal{N}(\{b_1\}) \cap \dots \cap \mathcal{N}(\{b_t\}) = \mathcal{N}(A) \cap \mathcal{N}(B)$   $\square$

6.3 Latin Collections

**Remark 6.3.1** Let  $A$  be a set of cells. A way of showing that no cell in  $A$  can be filled uniquely, would be to show that there exists a latin collection where each element of the collection has the same shape as  $A$ . A couple of examples of such collections are presented below.

The partial latin square below is a critical set  $C$  with unique completion next to it:

0	1	2	3	
1	2	3		
2	3			
3				

0	1	2	3	4
1	2	3	4	0
2	3	4	0	1
3	4	0	1	2
4	0	1	2	3

**Example 6.2** If  $x = (0,0;0)$  then  $C \setminus \{x\}$  has the following partial completion and full completions:

	1	2	3	
1	2	3		
2	3			1
3			1	2
		1	2	3

<b>0</b>	1	2	3	<b>4</b>
1	2	3	<b>4</b>	<b>0</b>
2	3	<b>4</b>	0	1
3	<b>4</b>	0	1	2
<b>4</b>	0	1	2	3

<b>4</b>	1	2	3	<b>0</b>
1	2	3	0	<b>4</b>
2	3	<b>0</b>	<b>4</b>	1
3	<b>0</b>	<b>4</b>	1	2
<b>0</b>	<b>4</b>	1	2	3

The partial latin square on the above left is in fact the nest of the entry  $x = (0,0;0)$ . The completions on the right are completions from  $C \setminus \{x\}$ . The bold entries in the two complete latin squares indicate the entries that have no unique completion, these sets of entries in fact form latin interchanges, with each set being the disjoint mate of the other. *Note that a pair of latin interchanges form a latin collection of size 2.*

**Example 6.3** If  $x = (0,1;1)$  then  $C \setminus \{x\}$  has the following partial completion and full completions:

0		2	3	
1	2	3		
2	3			
3				2
4			2	3

0	1	2	3	<b>4</b>
1	2	3	<b>4</b>	0
2	3	<b>4</b>	0	1
3	<b>4</b>	0	1	2
<b>4</b>	0	1	2	3

0	<b>4</b>	2	3	<b>1</b>
1	2	3	0	<b>4</b>
2	3	<b>4</b>	<b>1</b>	0
3	0	1	<b>4</b>	2
<b>4</b>	<b>1</b>	0	2	3

0	<b>4</b>	2	3	<b>1</b>
1	2	3	0	<b>4</b>
2	3	<b>1</b>	<b>4</b>	0
3	0	<b>4</b>	<b>1</b>	2
<b>4</b>	<b>1</b>	0	2	3

The set of non-empty cells in the partial latin square on the extreme left indicates the nest of the entry  $x = (0, 1; 1)$ . The three latin squares to the right are three completions from  $C \setminus \{x\}$ . These three full latin squares are necessary to show that each cell that is in bold type can be filled at least two ways. The three sets in bold make up a latin collection.

## 6.4 Influence of a set

**Theorem 6.12** *For any  $x \in C$ ,  $x \in \mathcal{I}(\{x\})$ , and for any  $y \in C$ , such that  $x \neq y, x \notin \mathcal{I}(\{y\})$ .*

**Proof.** By definition  $\mathcal{I}(\{x\}) = L \setminus \mathcal{N}(\{x\})$ . Since  $\mathcal{N}(\{x\})$  cannot contain  $x$ , therefore  $x \in \mathcal{I}(\{x\})$ . Also, by definition,  $C \setminus \{x\} \subset \mathcal{N}(\{x\})$ , and therefore  $y \in \mathcal{N}(\{x\})$ . That is,  $y \notin \mathcal{I}(\{x\})$ .

**Theorem 6.13** *The influence-set of a set  $B \subseteq C$  is the intersection of the influence-sets of its elements.*

**Proof.** If  $B \subseteq C$  then by definition

$$\begin{aligned} \mathcal{I}(B) &= L \setminus \cup\{\mathcal{N}(b) : b \in B\} \\ &= \cap\{L \setminus \mathcal{N}(b) : b \in B\} \\ &= \cap\{I(b) : b \in B\}. \end{aligned}$$

□

**Theorem 6.14** *For any two elements  $x, y \in C$  such that  $x \neq y$ ;  $x \notin \mathcal{I}(\{x, y\})$ , and  $y \notin \mathcal{I}(\{x, y\})$ .*

**Proof.** We have that  $\mathcal{I}(\{x, y\}) = \mathcal{I}(\{x\}) \cap \mathcal{I}(\{y\})$ . Since  $x \notin \mathcal{I}(\{y\})$  and  $y \notin \mathcal{I}(\{x\})$  we have the required result. □

**Corollary 6.15** *For any set  $A \subset C$ , with  $|A| \geq 2$ ,  $A \cap \mathcal{I}(A) = \emptyset$ .*

**Theorem 6.16** *Let  $L$  be a back-circulant latin square of order  $n$  with critical set  $C$ . If the lower (or upper) triangle in  $C$  has only one element, then there exists exactly one element having perfect influence.*

**Proof.** Suppose the lower triangle in  $C$  consists of only a single entry. Delete this entry. Then no empty entry can be uniquely completed.  $\square$

**Notation 1** *For ease of notation, we will denote the points on the boundaries of the two triangles by special symbols. In the upper triangle, let:*

1.  $\alpha = (0, 0; 0)$ ,
2.  $\alpha_1 = (r, 0; r)$ ,
3.  $\alpha_2 = (0, r; r)$ ,
4.  $\alpha_{\text{row}}$  be any entry in row 0, between  $\alpha$  and  $\alpha_1$ , exclusive,
5.  $\alpha_{\text{col}}$  be any entry in column 0, between  $\alpha$  and  $\alpha_2$ , exclusive,
6.  $\alpha_{\text{diag}}$  be any entry in the largest diagonal from lower left to upper right, in the upper triangle, between  $\alpha_1$  and  $\alpha_2$ , exclusive.

*In the lower triangle, let:*

1.  $\beta = (n - 1, n - 1; n - 2)$ ,
2.  $\beta_1 = (n - 1, r + 2; r + 1)$ ,
3.  $\beta_2 = (r + 2, n - 1; r + 1)$ ,
4.  $\beta_{\text{row}}$  be any entry in row  $n - 1$ , between  $\beta_1$  and  $\beta$ , exclusive,
5.  $\beta_{\text{col}}$  be any entry in column  $n - 1$ , between  $\beta_2$  and  $\beta$ , exclusive,
6.  $\beta_{\text{diag}}$  be any entry in the largest diagonal from the lower left to upper right, in the lower triangle, between  $\beta_1$  and  $\beta_2$ , exclusive.

We first calculate the nest of each of the six corner entries of the two triangles in  $C$ .

**Theorem 6.17** *Let  $L$  and  $C$  be the latin square and critical set discussed above. Then the six corner entries in the two triangles of  $C$ , have the following nests:*

1.  $\mathcal{N}(\alpha) = \{(u, v; u + v) : 1 < u, v \leq n - 1 \text{ and } 0 < u + v \leq r\} \cup C \setminus \{\alpha\}$ ,
2.  $\mathcal{N}(\beta) = \{(u, v; u + v) : 0 \leq u, v < n - 2 \text{ and } r < u + v < n - 2\} \cup C \setminus \{\beta\}$ ,

3.  $\mathcal{N}(\alpha_1) = \{(r+1, 0; r+1), \dots, (n-1, 0; n-1); (r, 1; r+1), \dots, (n-1, 1; 0); \dots; (2, r-1; r+1), \dots, (n-1, r-1; r-2)\} \cup C \setminus \{\alpha_1\},$
4.  $\mathcal{N}(\alpha_2) = \{(0, r+1; r+1), \dots, (0, n-1; n-1); (1, r; r+1), \dots, (1, n-1; 0); \dots; (r-1, 2; r+1), \dots, (r-1, n-1; r-2)\} \cup C \setminus \{\alpha_2\},$
5.  $\mathcal{N}(\beta_1) = (0, r+3; r+3), \dots, (n-3, r+3; r); (0, r+4; r+4), \dots, (n-4, r+4; r); \dots; (0, n-1; n-1), \dots, (r+1, n-1; r)\} \cup C \setminus \{\beta_1\},$
6.  $\mathcal{N}(\beta_2) = \{(r+3, 0; r+3), \dots, (r+3, n-3; r); (r+4, 0; r+4), \dots, (r+4, n-4; r); \dots; (n-1, 0; n-1), \dots, (n-1, r+1; r)\} \cup C \setminus \{\beta_2\}.$

**Proof.** Apply Lemmas 6.1, 6.2 and 6.3. □

**Corollary 6.18** *Let  $L$  and  $C$  be the latin square and critical set as given above, of order  $n$ . Then for some choices of  $((i, j; k) :$*

1.  $|\mathcal{N}(\alpha)| = \frac{1}{2}[(n-2)(n-1) + (r+1)(r+2)] - 1,$
2.  $|\mathcal{N}(\beta)| = \frac{1}{2}[(n-2)(n-1) + (n-r-2)(n-r-1)] - 1,$
3.  $|\mathcal{N}(\alpha_1)| = \frac{1}{2}[(n-r-2)(n+r-1) + 2(r+1)^2] - 1,$
4.  $|\mathcal{N}(\alpha_2)| = \frac{1}{2}[(n-r-2)(n+r-1) + 2(r+1)^2] - 1,$
5.  $|\mathcal{N}(\beta_1)| = (n-r-3)(n+r+1) + \frac{1}{2}[(n-r-2)(n-r-1) + 2(r+2)^2 - n(n-1)] - 1,$
6.  $|\mathcal{N}(\beta_2)| = (n-r-3)(n+r+1) + \frac{1}{2}[(n-r-2)(n-r-1) + 2(r+2)^2 - n(n-1)] - 1.$

In the case where  $n$  is odd, we have:

**Theorem 6.19** *Let  $L$  be a latin square of order  $n \geq 9$ , let  $r = \frac{n-3}{2}$  for  $n$  odd, and  $r = \frac{n-2}{2}$  for  $n$  even. Then the following pairs of entries have perfect influence:*

1.  $\{\alpha, \beta\},$
2.  $\{\alpha_1, \beta_1\},$
3.  $\{\alpha_2, \beta_2\}.$

**Proof.**

1. If  $\alpha$  is deleted then every diagonal below the main diagonal going from lower left to upper right, except the main diagonal and the one immediately below it, can be filled. Also, if  $\beta$  is deleted, then every diagonal, above the main diagonal going from lower left to upper right, except the main diagonal and the one immediately above it, can be filled. No diagonal above the main diagonal can intersect a diagonal below the main diagonal. (Indeed no two diagonals can intersect). That is,  $\mathcal{N}(\alpha) \cap \mathcal{N}(\beta) = C \setminus \{\alpha, \beta\}$ , and we're done. Thus the pair  $\{\alpha, \beta\}$  have perfect influence.
2. If  $\alpha_1$  is deleted then every column  $\lambda, 0 \leq \lambda \leq r - 1$ , can be filled. No other row, column or diagonal can be filled. If  $\beta_1$  is deleted, then every column  $\lambda, r + 3 \leq \lambda \leq n - 1$ , can be filled. No other row, column or diagonal can be filled. Here again, the influences of  $\alpha_1$  and  $\beta_1$  have no intersection outside of the critical set, so  $\mathcal{N}(\alpha_1) \cap \mathcal{N}(\beta_1) = C \setminus \{\alpha_1, \beta_1\}$ . That is, the pair  $\{\alpha_1, \beta_1\}$  have perfect influence.
3. If  $\alpha_2$  is deleted then every row  $\gamma, 0 \leq \gamma \leq r - 1$ , can be filled. No other row, column or diagonal can be filled. If  $\beta_2$  is deleted, then every row,  $\gamma, r + 3 \leq \gamma \leq n - 1$ , can be filled. No other row, column or diagonal can be filled. The influences of  $\alpha_2$  and  $\beta_2$  have no intersection outside of the critical set, and so  $\mathcal{N}(\{\alpha_2, \beta_2\}) = C \setminus \{\alpha_2, \beta_2\}$ . Thus the pair  $\{\alpha_2, \beta_2\}$  have perfect influence.

**Theorem 6.20** *Let  $L$  be a latin square of odd order, and let  $r = \frac{n-3}{2}$ . Then the following sets of three entries each have perfect influence:*

1.  $\{\alpha, \beta_{row}^* = (n - 1, s; s - 1), \beta_{col}^* = (t, n - 1; t - 1)\}$ , where  $r + 2 \leq s \leq n - 2$ , and  $3r - s + 2 \leq t \leq n - 2$ ,
2.  $\{\beta, \alpha_{row}^* = (0, s; s), \alpha_{col}^* = (t, 0; t)\}$ , where  $1 \leq s, t \leq r$  and  $s + t = r + 1$ ,
3.  $\{\alpha_1, \alpha_2, \beta_{diag}\}$ ,
4.  $\{\beta_1, \beta_2, \alpha_{diag}\}$ ,
5.  $\{\alpha, \alpha_1, \beta_{row}\}$ ,
6.  $\{\alpha, \alpha_2, \beta_{col}\}$ ,
7.  $\{\beta, \beta_1, \beta_{row}\}$ ,
8.  $\{\beta, \beta_2, \alpha_{col}\}$ .

**Proof.**

1. Suppose  $\alpha$  is deleted. Then each of the diagonals, below the main diagonal going from lower left to upper right, except the main diagonal, and the diagonal immediately below the main diagonal, can be filled. The other two entries deleted are from the lower triangle. Any diagonal in the upper triangle that is filled after the deletion of any of these entries cannot intersect with the set of elements completed after the deletion of  $\alpha$ , outside of  $C$ , and so we need not consider any such diagonal. From the results in Section 6.2, we need only check that row  $t + 1$ , and column  $s + 1$  do not intersect outside of  $C$ . But since the smallest sum of  $s$  and  $t$  is  $r + 1$ , and the largest is  $n - 2$ , we obtain the required result. That is, for the given  $s$  and  $t$  values,  $\mathcal{N}(\{\alpha, (n - 1, s; s - 1), (t, n - 1; t - 1)\}) = C \setminus \{\alpha, (n - 1, s; s - 1), (t, n - 1; t - 1)\}$ .
2. If  $\beta$  is deleted then each diagonal, above the main diagonal going from lower left to upper right, except the main diagonal, and the diagonal immediately above the main diagonal, can be filled. The other two entries are deleted from the upper triangle. Similar to the proof of (1.) above, we need only show that row  $t - 1$  and column  $s - 1$  intersect inside  $C$ . But this is tantamount to showing that  $0 \leq t - 1 + s - 1 \leq r$ , and this can be easily achieved from the inequalities defining  $s$  and  $t$  above.
3. If  $\alpha_1$  is deleted, then each of the columns  $\lambda, 0 \leq \lambda \leq r - 1$  can be completed. No other row, column or diagonal can be completed. If  $\alpha_2$  is deleted, then each of the rows  $\gamma, 0 \leq \gamma \leq r - 1$  can be completed. No other row, column or diagonal can be completed. The set of entries that can be completed after the deletion of both  $\alpha_1$  and  $\alpha_2$  is precisely the intersection of the sets of entries that can be completed after the deletion of each of these elements separately. This set is the region bounded by the diagonal above the main diagonal from lower left to upper right, (but excluding this diagonal), and column  $r$  and row  $r$ , again excluding these row and column. The entries that can be completed after deletion of  $\beta_{diag}$  will occur in rows below the row containing  $\beta_{diag}$  and in columns to the right of the column containing  $\beta_{diag}$ . This row and column cannot intersect the above set of filled entries discussed above, and so  $\mathcal{N}(\{\alpha_1, \alpha_2, \beta_{diag}\}) = C \setminus \{\alpha_1, \alpha_2, \beta_{diag}\}$ . Thus we have perfect influence.
4. The proof is similar to the proof of the above.

5. If  $\alpha$  is deleted then all the diagonals below the main diagonal going from the lower left to the upper right, except the main diagonal and the diagonal immediately below it, can be filled. If  $\alpha_1$  is deleted then each of the columns to the left of the column containing  $\alpha_1$  can be filled. These sets of filled entries intersect in a region to the left of the lower triangle of  $C$ , and below the main diagonal. If  $\beta_{row}$  is deleted then any filled entries will occur in the columns to the right of column containing this entry. Thus the intersection of this set of filled entries, and the set of entries filled on deletion of  $\alpha$ , occurs to the right of the lower triangle, and below the main diagonal. Clearly the two intersections do not themselves intersect. Thus  $\mathcal{N}(\{\alpha, \alpha_1, \beta_{row}\}) = C \setminus \{\alpha, \alpha_1, \beta_{row}\}$ .
6. The proof is similar to the above.
7. The proof is similar to the above.
8. The proof is similar to the above. □

**Theorem 6.21** *The following set of four entries has perfect influence:*

$$\{\alpha_{row}, \alpha_{col}, \beta_{row}, \beta_{col}\}$$

**Proof.** The set of entries that can be completed after the deletion of both the entries  $\alpha_{row}$  and  $\alpha_{col}$  will occur entirely above the main diagonal going from lower left to upper right. The set of entries that can be completed after the deletion of both the entries,  $\beta_{row}$  and  $\beta_{col}$ , will lie entirely below the diagonal going from lower left to upper right. Thus the two sets have an empty intersection, giving  $\mathcal{N}(\{\alpha_{row}, \alpha_{col}, \beta_{row}, \beta_{col}\}) = C \setminus \{\alpha_{row}, \alpha_{col}, \beta_{row}, \beta_{col}\}$ . Thus the set has perfect influence. □

**Theorem 6.22** *The strong box  $\mathcal{I}(C)$  is given by*

$$L \setminus \cup \{\mathcal{N}(\alpha), \mathcal{N}(\beta), \mathcal{N}(\alpha_1), \mathcal{N}(\alpha_2), \mathcal{N}(\beta_1), \mathcal{N}(\beta_2)\}$$

*and has size*

$$0 \leq \theta(C) \leq 6$$

*for  $n$  even, and*

$$0 \leq \theta(C) \leq 7$$

*for  $n$  odd.*



**Proof.** If  $\alpha$  is deleted then every diagonal below the main diagonal going from lower left to upper right can be completed, except the main diagonal and the diagonal immediately below it. This is the maximum number of diagonals that can be filled on deletion of any set from  $C$ . Also, if  $\beta$  is deleted, then we can fill all the diagonals above the main diagonal, excluding the main diagonal, and the one immediately above it. This is the maximum number of diagonals that can be filled above the main diagonal. Similarly, the deletion of the remaining entries gives the maximum number of rows, and columns that can be filled. Thus, the cells that are not in the union of the nests of these elements must form the influence of  $C$ . That is,  $\mathcal{I}(C) = L \setminus \cup\{\mathcal{N}(\alpha), \mathcal{N}(\beta), \mathcal{N}(\alpha_1), \mathcal{N}(\alpha_2), \mathcal{N}(\beta_1), \mathcal{N}(\beta_2)\}$ .

If  $r = n - 2$ , then  $C$  consists of only the upper triangle. It is easy to check that the union of the nests of the entries covers the whole latin square. Thus, there is no influence when all these entries are deleted. That is,  $\mathcal{I}(C) = \phi$ , so  $\theta(C) = 0$ .

Let  $n$  be even, and  $r = \frac{n-2}{2}$ . Then the union of the nests of the above entries covers all the columns except columns  $r$  to  $r + 2$ , all rows except rows  $r$  to  $r + 2$ , all the diagonals going from lower left to upper right, except those diagonals although the cells  $(n - 2, 0), (n - 1, 0), (n - 1, 1)$ . This union covers all entries, except the entries  $(r, r), (r, r + 1), (r, r + 2); (r + 1, r), (r + 1, r + 1); (r + 2, r + 2)$ . Thus  $\mathcal{I}(C) = (r, r), (r, r + 1), (r, r + 2); (r + 1, r), (r + 1, r + 1); (r + 2, r + 2)$ , and  $\theta(C) = 6$ .

Let  $n$  be odd, and  $r = \frac{n-3}{2}$ . Then the union of the nests of these entries covers all cells, except  $(r, r + 1), (r, r + 2); (r + 1, r), (r + 1, r + 1), (r + 1, r + 2); (r + 2, r), (r + 2, r + 1)$ . Thus,  $\mathcal{I}(C) = (r, r + 1), (r, r + 2); (r + 1, r), (r + 1, r + 1), (r + 1, r + 2); (r + 2, r), (r + 2, r + 1)$ , and  $\theta(C) = 7$ .  $\square$

## 6.5 A hierarchy of influence

Clearly,

**Theorem 6.23** *For any three elements  $x, y, z \in C$ ,  $x \longrightarrow y$  and  $y \longrightarrow z \implies x \longrightarrow z$ .*

**Example 6.4** Consider the critical set for the  $7 \times 7$  back-circulant latin square:

0	1	2				
1	2					
2						
						3
					3	4
				3	4	5

Let  $\alpha = (0, 0; 0), \alpha_1 = (0, 2; 2), \alpha_2 = (2, 0; 2), \alpha_{row} = (0, 1; 1), \alpha_{col} = (1, 0; 1), \alpha_{diag} = (1, 1; 2), \beta = (6, 6; 5), \beta_1 = (6, 4; 3), \beta_2 = (4, 6; 3), \beta_{row} = (6, 5; 4), \beta_{col} = (5, 6; 4), \beta_{diag} = (5, 5; 3)$ . The nests of these entries are as follows:

	1	2				
1	2					
2						1
					1	2
				1	2	3
			1	2	3	4
		1	2	3	4	5

Nest of  $\alpha$

0	1					
1	2					
2	3					
3	4					
4	5					3
5	6				3	4
6	0			3	4	5

Nest of  $\alpha_1$

0	1	2	3	4	5	6
1	2	3	4	5	6	0
						3
					3	4
				3	4	5

Nest of  $\alpha_2$

0		2				
1	2					
2						
3						2
4					2	3
5				2	3	4
6			2	3	4	5

Nest of  $\alpha_{row}$

0	1	2	3	4	5	6
	2					
2						
						2
					2	3
				2	3	4
			2	3	4	5

Nest of  $\alpha_{col}$

0	1	2	3	4	5	6
1						
2						
3						
4						3
5					3	4
6				3	4	5

Nest of  $\alpha_{diag}$

0	1	2	3	4		
1	2	3	4			
2	3	4				
3	4					
4						3
					3	4
				3	4	

Nest of  $\beta$

0	1	2			5	6
1	2				6	0
2					0	1
					1	2
					2	3
					3	4
					4	5

Nest of  $\beta_1$

0	1	2				
1	2					
2						
5	6	0	1	2	3	4
6	0	1	2	3	4	5

Nest of  $\beta_2$

0	1	2	3			6
1	2	3				0
2	3					1
3						2
						3
					3	4
				3		5

Nest of  $\beta_{row}$

0	1	2	3			
1	2	3				
2	3					
3						
						3
					3	
6	0	1	2	3	4	5

Nest of  $\beta_{col}$

0	1	2				6
1	2					0
2						1
						2
						3
						4
6	0	1	2	3	4	5

Nest of  $\beta_{diag}$

From inspection of the nests, we find that the following sets have perfect influence:

1.  $\{\alpha, \beta\}$ ;
2.  $\{\alpha, \alpha_1\}, \{\alpha, \alpha_2\}, \{\alpha, \alpha_{diag}\}, \{\alpha_1, \alpha_2\}, \{\alpha_1, \alpha_{col}\}, \{\alpha_2, \alpha_{row}\}$ ;
3.  $\{\beta, \beta_1\}, \{\beta, \beta_2\}, \{\beta, \beta_{diag}\}, \{\beta_1, \beta_2\}, \{\beta_1, \beta_{col}\}, \{\beta_2, \beta_{row}\}$ ;
4.  $\{\alpha_1, \beta_1\}, \{\alpha_2, \beta_2\}$ ;
5.  $\{\alpha, \beta_{row}, \beta_{col}\}; \{\beta, \alpha_{row}, \alpha_{col}\}$ ;

6.  $\{\alpha_{row}, \alpha_{col}, \beta_{row}, \beta_{col}\}.$

From these perfect influence sets we have that  $\alpha \rightsquigarrow \alpha_{row}, \alpha \rightsquigarrow \alpha_{col}$ , and  $\beta \rightsquigarrow \beta_{row}, \beta \rightsquigarrow \beta_{col}$ . That is,  $\alpha$  is more influential than  $\alpha_{row}$  and  $\alpha_{col}$ , etc. In terms of perfect influence, it means, for example, that given any set  $A$  of non-perfect influence, if  $\alpha_{row}A$  has perfect influence, then  $\alpha A$  will also have perfect influence. Similarly, if  $\{\alpha_{row}, \alpha_{col}\} \cup A$  has perfect influence, then  $\alpha A$  will also have perfect influence.

The table below is obtained by replacing each entry in  $C$  with its influence:

29	30	29				
30	30					
29						
						29
					30	30
				29	30	29

That is,

$$\begin{aligned}\theta(\alpha) &= \theta(\beta) = \theta(\alpha_1) = \theta(\alpha_2) = \theta(\beta_1) = \theta(\beta_2) = 29, \\ \theta(\alpha_{row}) &= \theta(\alpha_{col}) = \theta(\alpha_{diag}) = \theta(\beta_{row}) = \theta(\beta_{col}) = \theta(\beta_{diag}) = 30.\end{aligned}$$

So

$$[\alpha] = \{\alpha, \beta, \alpha_1, \alpha_2, \beta_1, \beta_2\} \text{ and } [\alpha_{row}] = \{\alpha_{row}, \alpha_{col}, \beta_{row}, \beta_{col}\}.$$

That is, the index of  $C$  is 2.

A secret sharing scheme based on a critical set of a latin square would have each triple associated with the critical set distributed as shares. If the size of the critical set is  $t$  then this is a  $t$ -out-of- $t$  scheme, see for example [38].

In general the critical sets which have been looked at are unions of two triangles of triples, one in the upper left hand corner, denoted  $UL$ , and the other in the lower right hand corner, denoted  $LR$ . In such a case the scheme will be compartmentalised, as elements of both compartments are needed to complete the latin square. It has been shown that some triples have more influence than others. Thus the scheme will not only be compartmentalised, but hierarchical as well.

## Critical Sets in Modified-Two Back-Circulants

This chapter is based on [31]. At least 95 percent of the material in this chapter is due to the author of this thesis.

### 7.1 Introduction

In the previous chapter, different types of back-circulant latin squares namely,  $BC(2, n)$ ,  $BCFC(2, n)$ ,  $MBC(2, n)$ ,  $BC(2^t)$  and  $BC(mn)$  were studied. General constructions of uniquely completable sets for  $MBC(2, n)$  and  $BCFC(2, n)$  were also given. These uniquely completable sets have been studied to evaluate the change in size of the critical set or, in some cases, the smallest known uniquely completable set with the constructions of Gower et al..

In this chapter, focus is on the identification of latin interchanges in modified-two back-circulant latin squares containing odd order subsquares. These latin interchanges will be used to identify critical sets in  $MBC(2, n)$ . Latin interchanges have already been used to establish the existence of critical sets in back-circulant latin squares, to identify subsquare free latin squares and to investigate the intersection sizes of latin squares. Latin interchanges have been studied by Keedwell [73], Cooper, Donovan and Seberry in [37], Gower [60] and Cooper, Donovan and Gower in [39].

A modified-two back-circulant  $MBC(2, n)$  latin square has the following form:

$$\begin{bmatrix} AR & BR \\ BR & SA \end{bmatrix} = \begin{bmatrix} AR & UpperBR \\ LowerBR & SA \end{bmatrix}$$

where, for latin subsquares,  $A$  and  $B$  represent circulant matrices,  $AR$  or  $BR$  for back-circulant matrices and  $SA$  for the back-circulant matrix which has the first row of  $A$  reversed.  $csMBC(2, n)$  denotes critical set of modified-two back-circulant latin

squares. So for order 6, we have  $MBC(2, 3)$  which has the first row of the right hand bottom subsquare reversed:

**Example 7.1** A modified-two back-circulant  $MBC(2,3)$  latin square

1	2	3	4	5	6
2	3	1	5	6	4
3	1	2	6	4	5
4	5	6	3	2	1
5	6	4	2	1	3
6	4	5	1	3	2

Let  $N = \{1, 2, \dots, n\}$  and  $I = \{(i, j; k) | i, j, k \in N\}$  be a partial latin square of order  $n$ . Then  $|I|$  is said to be the *size* of the partial latin square and the set of cells  $\{(i, j) | i, j, k \in I, \exists k \in N\}$  is said to determine the *shape* of  $I$ . Let  $I$  and  $I'$  be two partial latin squares of order  $n$ , with the same shape. Then  $I$  and  $I'$  are said to be *mutually balanced* if the entries in each row (and column) of  $I$  are the same as those in the corresponding row (and column) of  $I'$ . They are said to be *disjoint* if no cell in  $I'$  contains the same entry as the corresponding cell of  $I$ . Given two partial latin squares  $I$  and  $I'$  of order  $n$ , of the same size and shape, with the property that  $I$  and  $I'$  are disjoint and mutually balanced,  $I$  is said to be a latin interchange and  $I'$  is said to be *disjoint mate* of  $I$ .

**Example 7.2** Following is an example of a latin interchange  $I_{(x,y)}$  and its disjoint mate  $I'_{(x,y)}$  for  $MBC(2,3)$  given in Example 7.1:

$I_{(1,1)} =$ 

5				1	
1				5	

$I'_{(1,1)} =$ 

1				5	
5				1	

The latin interchange  $I_{(1,1)}$  can also be represented by the set  $\{(1,1;5), (1,5;1), (5,1;1), (1,1;5)\}$ . The disjoint mate  $I'_{(1,1)}$  can be represented by the set  $\{(1,1;1), (1,5;5), (5,1;5), (1,1;1)\}$ .

Latin interchanges are important when studying critical sets in latin squares. A critical set is a partial latin square which is contained in precisely one latin square

and such that if one removes any entry from the partial latin square, then what is left is contained in at least two latin squares. To prove that a partial latin square  $C$  is critical, one must show that for each element of  $C$  there exists a latin interchange which intersects the critical set in that entry alone.

**Example 7.3** Take the following partial latin square  $C$  from the modified-two back-circulant  $\text{MBC}(2,3)$  given in Example 7.1:

1	2	3	4		
2	3	1			
3	1				5
4			3		
		5			2

It is easy to show that this partial latin square is uniquely contained in the latin square given in Example 7.1. However, to complete the proof that  $C$  is critical, one must show that if one removes any element of  $C$ , then what is left is contained in two latin squares. This is equivalent to showing that for each element of the critical set there exists a latin interchange which intersects the critical set in that element alone.

Take for example the entry 1 in row 1 and column 1 of  $C$  above. The latin interchange in example 7.2 intersects the partial latin square in this entry alone. It is easy to see that if this element is removed, the remaining partial latin square is contained in the following two latin squares:

1	2	3	4	5	6
2	3	1	5	6	4
3	1	2	6	4	5
4	5	6	3	2	1
5	6	4	2	1	3
6	4	5	1	3	2

5	2	3	4	1	6
2	3	1	5	6	4
3	1	2	6	4	5
4	5	6	3	2	1
1	6	4	2	5	3
6	4	5	1	3	2

Therefore one way to find critical sets in a given latin square is to identify all latin interchanges which are contained in that latin square and then find the partial latin square which minimally covers the latin interchanges.

**Example 7.4** The following partial square taken from the modified-two back-circulant  $\text{MBC}(2,3)$  latin square given in Example 7.1 is a critical set:

$$\begin{bmatrix} \textit{parAR} & \textit{parBR} \\ \textit{parBR} & \textit{parSA} \end{bmatrix} =$$

1	2	3	4		
2	3	1			
3	1				5
4			3		
		5			2

**Proof.** From Remark 5.3.2 in Chapter 5, we know that the subsquare  $BR$  can only be completed by elements  $n+1, \dots, 2n$ , hence if  $\textit{parBR}$  is a critical set, we can uniquely complete to the subsquare  $BR$  using elements  $n+1, \dots, 2n$ . Similarly if  $\textit{parSA}$  is a critical set, we can uniquely complete to the subsquare  $SA$  using elements  $1, \dots, n$ . Finally we note again, that  $\textit{parAR}$  will be completed to the subsquare  $AR$  only using elements 1 to  $n$ .

We know that  $\textit{parBR}$  and  $\textit{parSA}$  are critical sets of back-circulant latin square of order 3 as proved by Cooper, Donovan and Seberry in [37]. Latin interchanges and disjoint mates are given for each entry in  $\textit{parAR}$ . So each entry in this partial latin square has a latin interchange hence it is a critical set:

$I_{(1,1)} =$

5				1	
1				5	

$I'_{(1,1)} =$

1				5	
5				1	

$I_{(1,2)} =$

	5			2	
	2			5	

$I'_{(1,2)} =$

	2			5	
	5			2	

$I_{(1,3)} =$

		6			3
		4		2	
		2		1	6
		3		4	1

$I'_{(1,3)} =$

		3			6
		2		4	
		6		2	1
		4		1	3



$I_{(2,1)} =$

5			2		
2			5		

$I'_{(2,1)} =$

2			5		
5			2		

$I_{(2,2)} =$

	6			3	
		4		2	
		2		6	
	4	6			
	3			4	

$I'_{(2,2)} =$

	3			6	
		2		4	
		6		2	
	6	4			
	4			3	

$I_{(2,3)} =$

		6		1	
		4		2	
		2		6	
		1		4	

$I'_{(2,3)} =$

		1		6	
		2		4	
		6		2	
		4		1	

$I_{(3,1)} =$

6		4	2	3	
		2	1	4	
3			6	1	

$I'_{(3,1)} =$

3		2	6	4	
		4	2	1	
6			1	3	

$I_{(3,2)} =$

	6	4	2	1	
	1	2	6	4	

$I'_{(3,2)} =$

	1	2	6	4	
	6	4	2	1	

□

## 7.2 A family of latin interchanges

Take a partial modified-two back-circulant  $MBC(2, n) = MBC(2, 2m + 1)$  latin square as follows:

$$\begin{bmatrix} parAR & csBR \\ csBR & csSA \end{bmatrix}$$

where  $parAR$  is the partial subsquare of  $AR$ ,  $csBR$  is the critical set of  $BR$  and  $csSA$  is the critical set of  $SA$ . The above partial square  $MBC(2, 2m + 1)$  is given in generalised form in Appendix B. The partial subsquare  $parAR$  is further subdivided in the following figures where \*’s are unknown contents in the cells:

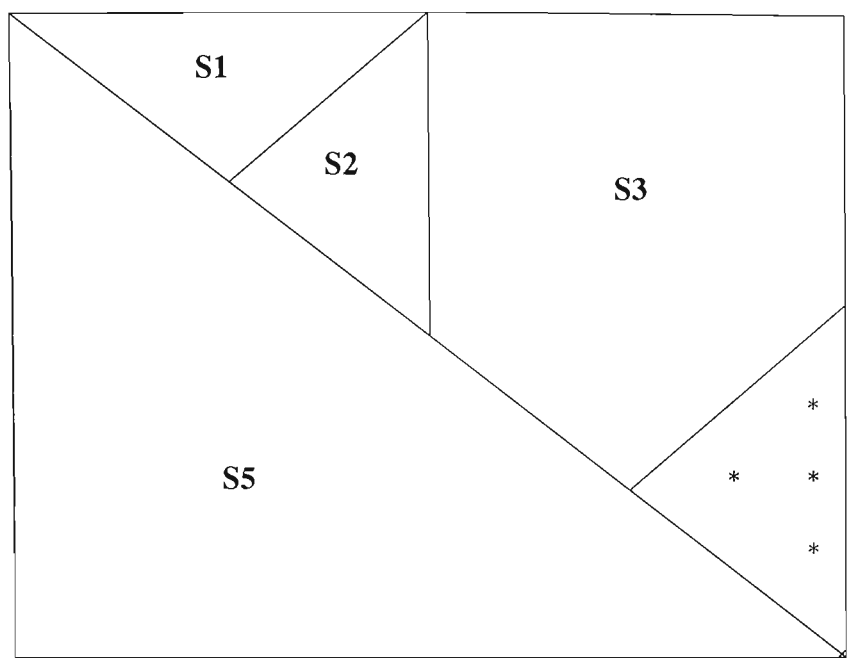


Figure 7.1: Sub-division of partial subsquare AR for  $m < 4$

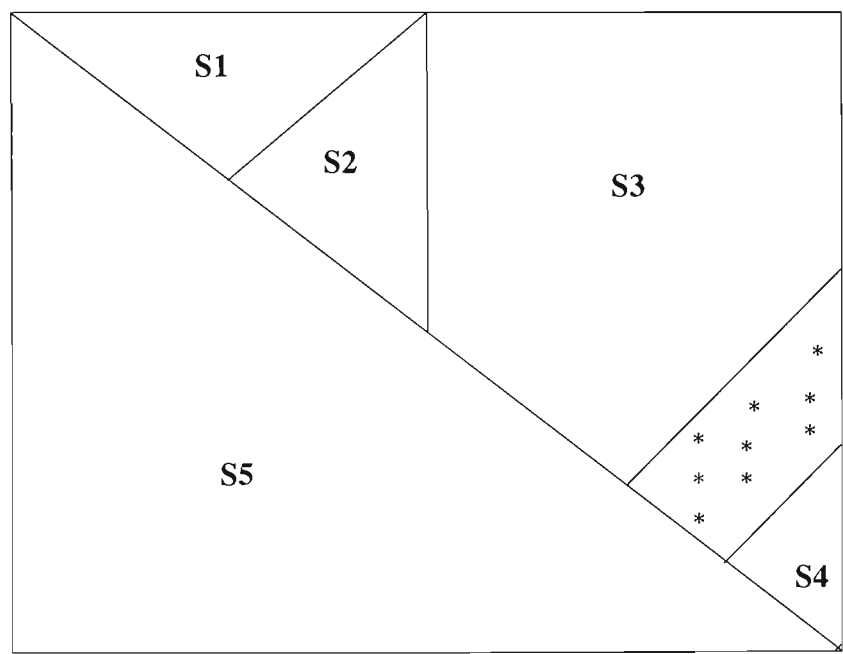


Figure 7.2: Sub-division of partial subsquare AR for  $m \geq 4$

The coordinates of S1, S2, S3, S4 and S5 are given in the following figures. It is noted that S1, S3, S4 have one element in the corner when  $m$  is even and two elements in the corner when  $m$  is odd, whereas S2 has one element in the corner when  $m$  is odd and two elements in the corner when  $m$  is even.

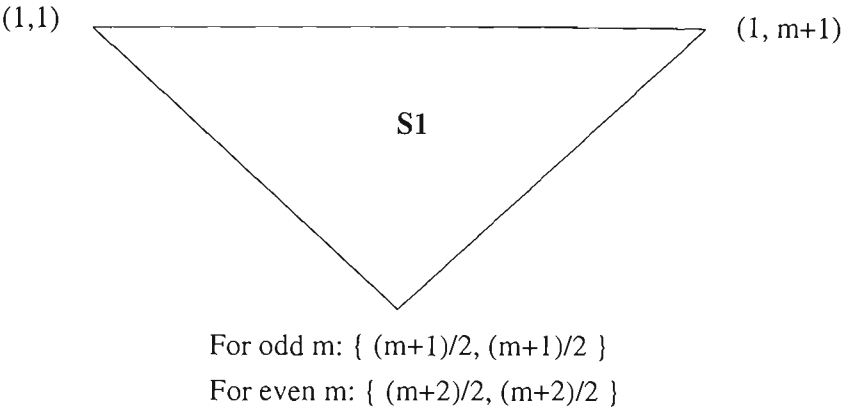


Figure 7.3: Coordinates of the partial subsquare S1

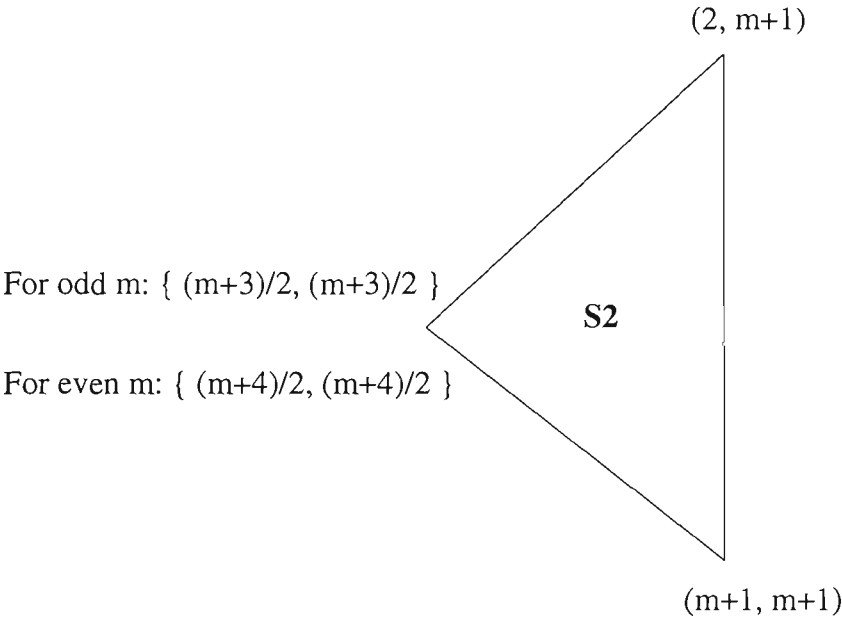


Figure 7.4: Coordinates of the partial subsquare S2

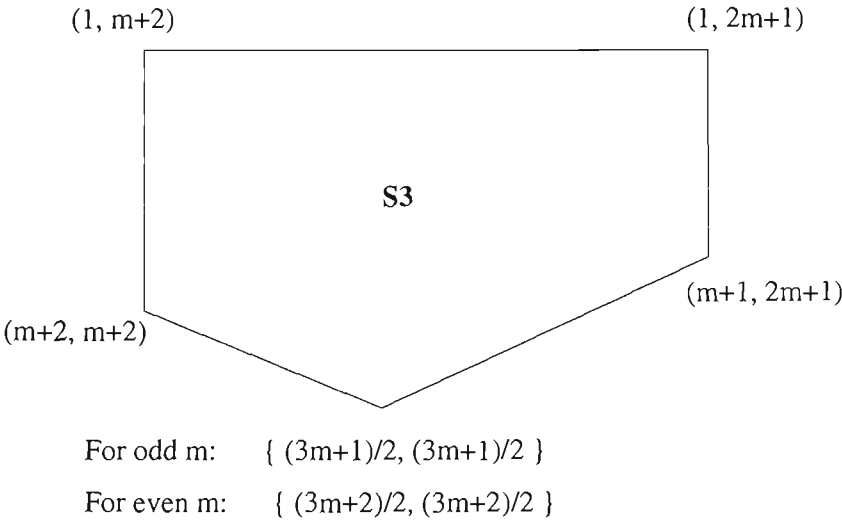


Figure 7.5: Coordinates of the partial subsquare S3

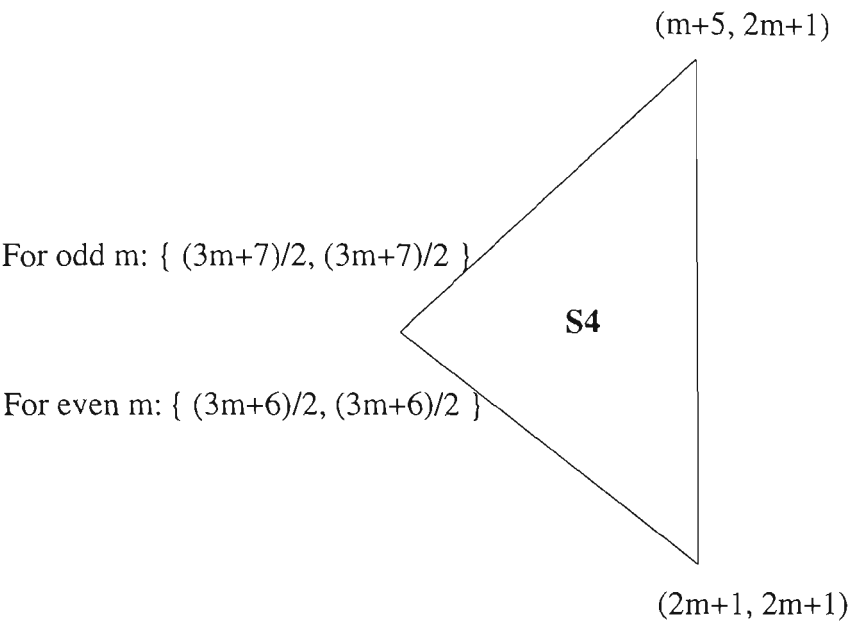


Figure 7.6: Coordinates of the partial subsquare S4

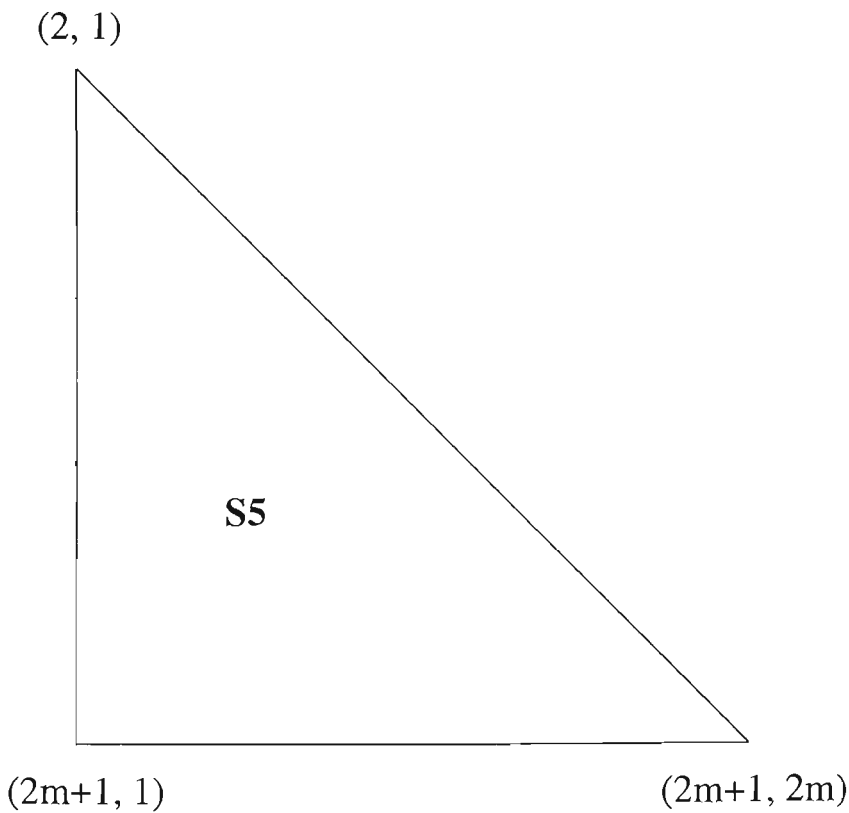


Figure 7.7: Coordinates of the partial subsquare S5



$$I'_{(1,3)} =$$

		3								11			
		11								3			

**Lemma 7.2** The coordinates of the cells  $\{(x, y)|x, y \in N\}$  in partial latin square  $S_2$  given in Figure 7.4 are:

$$x = \{m + 3 - y, \dots, y\} \text{ and } y = \{\frac{m+3}{2}, \dots, m + 1\} \text{ for odd order } m;$$

$$x = \{m + 3 - y, \dots, y\} \text{ and } y = \{\frac{m+4}{2}, \dots, m + 1\} \text{ for even order } m.$$

There exists a latin interchange  $I_{x,y}$  for every element  $\{(x, y; z)|x, y, z \in N\}$  in  $S_2$ .

**Proof.** Take a typical element  $(x, y; z)$  in  $S_2$  and construct the latin interchange  $I_{x,y}$  as follows: The two elements of  $I_{x,y}$  in the  $AR$  subsquare are

$$(x, y; 4m+2)$$

$$(2m+1, m+2; 2m+2)$$

The two elements of  $I_{x,y}$  in the  $UpperBR$  subsquare are

$$(x, 4m+3-x; x+y-1 \bmod n)$$

$$(2m+1, 2m+3; m+1)$$

The elements of  $I_{x,y}$  in the  $LowerBR$  subsquare are

$$(3m+1, m+2; m+1)$$

$$(4m+4-y, y; x+y-1 \bmod n)$$

For  $i = 1, 2, \dots, m+2-y$

$$(3m+1+i, m+2-i; 2m+2)$$

$$(3m+1+i, m+3-i; 4m+2)$$

Next  $i$

For the elements of  $I_{x,y}$  in the  $SA$  subsquare, we proceed as follows:

Let  $K_{Lt}$  be the top left element of the latin interchange in  $SA$ .

Let  $K_{Lb}$  be the bottom left element of the latin interchange in  $SA$ .

Let  $K_{Rt}$  be the top right element of the latin interchange in  $SA$ .

Let  $K_{Rb}$  be the bottom right element of the latin interchange in  $SA$ . So every latin interchange in  $SA$  has at least following four elements:

$$K_{Lt} = (3m+1, 2m+3; y-2)$$

$$K_{Lb} = (4m+4-y, 2m+3; 2m+2)$$

$$K_{Rt} = (3m+1, 4m+3-x; 4m+2)$$

$$K_{Rb} = (4m+4-y, 4m+3-x; x+m+2 \bmod n)$$

Let  $TC$  be the number of columns between  $K_{Lt}$  and  $K_{Rt}$ , then

$$\begin{aligned} TC &= (\text{y coordinate of } K_{Rt}) - (\text{y coordinate of } K_{Lt}) - 1 \\ &= (4m+3-x) - (2m+3) - 1 \\ &= 2m-1-x \end{aligned}$$

Let  $R$  be the number of rows between  $K_{Lt}$  and  $K_{Lb}$ , then

$$\begin{aligned} R &= (\text{x coordinate of } K_{Lb}) - (\text{x coordinate of } K_{Lt}) - 1 \\ &= (4m+4-y) - (3m+1) - 1 \\ &= m+2-y \end{aligned}$$

Let  $IC$  be the number of columns containing latin interchange between  $K_{Lt}$  and  $K_{Rt}$ .

If  $TC - R = 0$ , then Pattern P0

latin interchange consists of  $K_{Lt}$ ,  $K_{Rt}$ ,  $K_{Lb}$  and  $K_{Rb}$  elements only.

Else If  $\frac{TC+1}{R+1}$  is an integer, then Pattern P1

$$IC = \frac{TC-R}{R+1}$$

For  $i = 1, 2, \dots, IC$

$$K_{Mt_i} = (3m+1, 2m+3+i(R+1); y-2-i(R+1))$$

$$K_{Mb_i} = (4m+4-y, 2m+3+i(R+1); y-2-(i-1)(R+1))$$

Next  $i$

Else If  $\frac{TC}{R+1}$  is even, then Pattern P2

$$IC = \frac{TC}{R+1}$$

If  $IC-2 = 0$ , then

$$K_{Mt} = (3m+1, 3m-y+7; x+m+2 \bmod n)$$

$$K_{Mb} = (4m+4-y, 3m-y+6; y-2 \bmod n)$$

For  $i = 1, 2, \dots, R+1$

$$K_{M1_i} = (3m+i, 3m-y+6; y-2-i \bmod n)$$

$$K_{M2_i} = (3m+i+1, 3m-y+7; y-2-i \bmod n)$$

Next  $i$

Else

For  $i = 1, 2, \dots, \frac{IC-2}{2}$

$$K_{Lt_i} = (3m+1, 2m+3+i(R+1); y-2-i(R+1) \bmod n)$$

$$K_{Lb_i} = (4m+4-y, 2m+3+i(R+1); y-2-(i-1)(R+1) \bmod n)$$

Next  $i$

$$K_{Mt} = (3m+1, 3m-y+7+(\frac{IC-2}{2})(R+1); x+m+2+(\frac{IC-2}{2})(R+1) \bmod n)$$

$$K_{Mb} = (4m+4-y, 3m-y+6+(\frac{IC-2}{2})(R+1); y-2-(\frac{IC-2}{2})(R+1) \bmod n)$$



**For**  $i = 1, 2, \dots, R+1$

$$K_{M1_i} = (3m+i, 3m-y+6+(\frac{IC-2}{2})(R+1); y-2-(\frac{IC-2}{2})(R+1)-i \bmod n)$$

$$K_{M2_i} = (3m+i+1, 3m-y+7+(\frac{IC-2}{2})(R+1); y-2-(\frac{IC-2}{2})(R+1)-i \bmod n)$$

**Next**  $i$

**For**  $i = 1, 2, \dots, \frac{IC-2}{2}$

$$K_{Rt_i} = (3m+1, 4m+3-x-i(R+1); x+m+2+(i-1)(R+1) \bmod n)$$

$$K_{Rb_i} = (4m+4-y, 4m+3-x-i(R+1); x+m+2+i(R+1) \bmod n)$$

**Next**  $i$

**End If**

**Else If**  $\frac{TC}{R+1}$  is odd, then

Pattern P3

$$IC = \frac{TC}{R+1} + R$$

**If**  $IC-R-1 = 0$ , then

**For**  $i = 1, 2, \dots, R+1$

$$K_{M1_i} = (4m+4-y-i, 2m+3+i; y-3)$$

$$K_{M2_i} = (4m+4-y-(i-1), 2m+3+i; y-2)$$

**Next**  $i$

**Else**

**For**  $i = 1, 2, \dots, \frac{IC-R-3}{2}$

$$K_{Lt_i} = (3m+1, 2m+3+i(R+1); y-2-i(R+1) \bmod n)$$

$$K_{Lb_i} = (4m+4-y, 2m+3+i(R+1); y-2-(i-1)(R+1) \bmod n)$$

**Next**  $i$

**For**  $i = 1, 2, \dots, R+1$

$$K_{M1_i} = (4m+4-y-i, 2m+3+i+(\frac{IC-2}{2})(R+1); y-3-(\frac{IC-2}{2})(R+1))$$

$$K_{M2_i} = (4m+4-y-(i-1), 2m+3+i+(\frac{IC-2}{2})(R+1); y-2-(\frac{IC-2}{2})(R+1))$$

**Next**  $i$

**For**  $i = 1, 2, \dots, \frac{IC-R-3}{2}$

$$K_{Rt_i} = (3m+1, 4m+3-x-i(R+1); x+m+2+(i-1)(R+1) \bmod n)$$

$$K_{Rb_i} = (4m+4-y, 4m+3-x-i(R+1); x+m+2+i(R+1) \bmod n)$$

**Next**  $i$

**End If**

**Else If**  $\frac{TC}{R}$  is even, then

Pattern P4

**If**  $\frac{TC}{R}-2 = 0$ , then

$$K_{Mt} = (3m+1, 3m-y+6; x+m+2 \bmod n)$$

$$K_{Mb} = (4m-y+4, 3m-y+5; y-2 \bmod n)$$

**For**  $i = 1, 2, \dots, R$

$$K_{M1_i} = (3m+1+i, 3m-y+5; y-2-i \bmod n)$$

$$K_{M2_i} = (3m+1+i, 3m-y+6; y-1-i \bmod n)$$

Next i

Else

$$IC = \frac{TC}{R} - 1$$

For i = 1, 2, ...,  $\frac{IC-2}{2}$

$$K_{Lt_i} = (3m+1, 2m+3+i(R+1); y-2-i(R+1) \bmod n)$$

$$K_{Lb_i} = (4m+4-y, 2m+3+i(R+1); y-2-(i-1)(R+1) \bmod n)$$

Next i

$$K_{Mt} = (3m+1, 3m-y+6+(\frac{IC-2}{2})(R+1); x+m+2+(\frac{IC-2}{2})(R+1) \bmod n)$$

$$K_{Mb} = (4m+4-y, 3m-y+5+(\frac{IC-2}{2})(R+1); y-2-(\frac{IC-2}{2})(R+1) \bmod n)$$

For i = 1, 2, ..., R

$$K_{M1_i} = (3m+1+i, 3m-y+5+(\frac{IC-2}{2})(R+1); y-2-i-(\frac{IC-2}{2})(R+1) \bmod n)$$

$$K_{M2_i} = (3m+1+i, 3m-y+6+(\frac{IC-2}{2})(R+1); y-1-i-(\frac{IC-2}{2})(R+1) \bmod n)$$

Next i

For i = 1, 2, ...,  $\frac{IC-2}{2}$

$$K_{Rt_i} = (3m+1, 4m+3-x-i(R+1); x+m+2+(i-1)(R+1) \bmod n)$$

$$K_{Rb_i} = (4m+4-y, 4m+3-x-i(R+1); x+m+2+i(R+1) \bmod n)$$

Next i

End If

Else If  $\frac{TC-1}{R+1}$  is even for  $R \geq 3$ , then

Pattern P5

$$IC = \frac{TC-1}{R+1}$$

If  $IC - 2 = 0$ , then

$$K_{Mt} = (3m+1, 2m+6+R; x+m+2 \bmod n)$$

$$K_{Mb} = (4m+4-y, 2m+4+R; y-2)$$

For i = 0, 1, ...,  $\frac{R-1}{2}$

$$K_{M1_i} = (3m+1+2i, 2m+4+R; y-2(i+2) \bmod n)$$

$$K_{M2_i} = (3m+3+2i, 2m+6+R; y-2(i+2) \bmod n)$$

Next i

Else

For i = 1, 2, ...,  $\frac{IC-2}{2}$

$$K_{Lt_i} = (3m+1, 2m+3+i(R+1); y-2-i(R+1) \bmod n)$$

$$K_{Lb_i} = (4m+4-y, 2m+3+i(R+1); y-2-(i-1)(R+1) \bmod n)$$

Next i

$$K_{Mt} = (3m+1, 2m+6+R+(\frac{IC-2}{2})(R+1); x+m+2+(\frac{IC-2}{2})(R+1) \bmod n)$$

$$K_{Mb} = (4m+4-y, 2m+4+R+(\frac{IC-2}{2})(R+1); y-2-(\frac{IC-2}{2})(R+1) \bmod n)$$

**For**  $i = 0, 1, \dots, \frac{R-1}{2}$

$$K_{M1_i} = (3m+1+2i, 2m+4+R+(\frac{IC-2}{2})(R+1); y-2(i+2)-(\frac{IC-2}{2})(R+1) \bmod n)$$

$$K_{M2_i} = (3m+3+2i, 2m+6+R+(\frac{IC-2}{2})(R+1); y-2(i+2)-(\frac{IC-2}{2})(R+1) \bmod n)$$

**Next**  $i$

**For**  $i = 1, 2, \dots, \frac{IC-2}{2}$

$$K_{Rt_i} = (3m+1, 4m+3-x-i(R+1); x+m+2+(i-1)(R+1) \bmod n)$$

$$K_{Rb_i} = (4m+4-y, 4m+3-x-i(R+1); x+m+2+i(R+1) \bmod n)$$

**Next**  $i$

**End If**

**Else If**  $\frac{TC-1}{R+1}$  is odd for  $R \geq 3$ , then

Pattern P6

$$IC = \frac{TC+R}{R+1}$$

**If**  $IC - 2 = 0$ , then

$$K_{M1_1} = (3m+1+\frac{R+1}{2}, 2m+3+\frac{R+1}{2}; m+x+2)$$

$$K_{M1_2} = (4m+4-y, 2m+3+\frac{R+1}{2}; y-2)$$

$$K_{M2_1} = (3m+1, 2m+5+R; x+m+2)$$

$$K_{M2_2} = (3m+1+\frac{R+1}{2}, 2m+5+R; y-2)$$

**Else**

**For**  $i = 1, 2, \dots, \frac{IC-2}{2}$

$$K_{Lt_i} = (3m+1, 2m+3+i(R+1); y-2-i(R+1) \bmod n)$$

$$K_{Lb_i} = (4m+4-y, 2m+3+i(R+1); y-2-(i-1)(R+1) \bmod n)$$

**Next**  $i$

$$K_{M1_1} = (3m+1+\frac{R+1}{2}, 2m+3+(IC-1)\frac{R+1}{2}; m+x+2+(\frac{IC-2}{2})(R+1) \bmod n)$$

$$K_{M1_2} = (4m+4-y, 2m+3+(IC-1)\frac{R+1}{2}; y-2-(\frac{IC-2}{2})(R+1) \bmod n)$$

$$K_{M2_1} = (3m+1, 2m+3+IC\frac{R+1}{2}; m+x+2+(\frac{IC-2}{2})(R+1) \bmod n)$$

$$K_{M2_2} = (3m+1+\frac{R+1}{2}, 2m+3+IC\frac{R+1}{2}; y-2-(\frac{IC-2}{2})(R+1) \bmod n)$$

**For**  $i = 1, 2, \dots, \frac{IC-2}{2}$

$$K_{Rt_i} = (3m+1, 4m+3-x-i(R+1); x+m+2+(i-1)(R+1) \bmod n)$$

$$K_{Rb_i} = (4m+4-y, 4m+3-x-i(R+1); x+m+2+i(R+1) \bmod n)$$

**Next**  $i$

**End If**

**Else If**  $\frac{TC-1}{R}$  is an integer, then

Pattern P7

**For**  $i = 1, 2, \dots, \frac{TC-R-1}{2R}$

$$K_{Lt_i} = (3m+1, 2m+3+i(R+1); y-2-i(R+1) \bmod n)$$

$$K_{Lb_i} = (4m+4-y, 2m+3+i(R+1); y-2-(i-1)(R+1) \bmod n)$$

Next i

For  $i = 1, 2, \dots, R$

$$K_{M1_i} = (3m+1+i, 4m+2-R-x-i; y-3-R)$$

$$K_{M2_i} = (3m+1+i, 4m+3-R-x-i; y-2-R)$$

Next i

For  $i = 1, 2, \dots, \frac{TC-R-1}{2R}$

$$K_{Rt_i} = (3m+1, 4m+3-x-i(R+1); x+m+2+(i-1)(R+1) \bmod n)$$

$$K_{Rb_i} = (4m+4-y, 4m+3-x-i(R+1); x+m+2+i(R+1) \bmod n)$$

Next i

End If



**Example 7.6** Following are some of the instances of the latin interchanges  $I_{(x,y)}$  and their disjoint mates  $I'_{(x,y)}$  in  $S_2$  for each pattern given in Lemma 7.2:

An instance of pattern P0 in MBC2.7

[illegible][illegible]

An instance of pattern P1 in MBC2.7

[illegible][illegible]

An instance of pattern P2 in MBC2.11

[illegible]

[illegible]

An instance of pattern P3 in MBC2.7

[illegible][illegible]

An instance of pattern P4 in MBC2.11

[illegible][illegible]





An instance of pattern P6 in MBC2.11

[illegible][illegible]



**Proof.** Take a typical element  $(x, y; z)$  in  $S3$  and construct the latin interchange  $I_{x,y}$  as follows: The two elements of  $I_{x,y}$  in the  $AR$  subsquare are

$$(x, y; 4m+2)$$

$$(3m+3-y, y; 2m+2)$$

The two elements of  $I_{x,y}$  in the *UpperBR* subsquare are

$$(x, 4m+3-x; x+y-1 \bmod n)$$

$$(3m+3-y, m+1+y; m+1)$$

The two elements of  $I_{x,y}$  in the *LowerBR* subsquare are

$$(4m+3-y, y; m+1)$$

$$(4m+4-y, y; x+y-1 \bmod n)$$

At this point it is noted that the elements in rows  $x$  and  $3m+3-y$  have not altered but only the positions of the elements. Also the elements of the column  $y$  have not altered only their positions. For the elements of  $I_{x,y}$  in the  $SA$  subsquare, we have

If  $3m+3-x-y = 1$ , then

$$(4m+3-y, 4m+3-x; 4m+2)$$

$$(4m+4-y, 4m+3-x; 2m+2)$$

Else

$$(4m+3-y, m+1+y; m)$$

$$(4m+4-y, m+1+y; 2m+2)$$

$$(4m+3-y, 4m+3-x; 4m+2)$$

$$(4m+4-y, 4m+3-x; x+y-2m-1 \bmod n)$$

If  $3m+3-x-y > 2$ , then

For  $i = 1, 2, \dots, 3m+1-x-y$

$$(4m+3-y, m+1+y+i; m-i \bmod n)$$

$$(4m+4-y, m+1+y+i; m-i+1 \bmod n)$$

Next  $i$

End If

End If

It is noted if  $3m+3-x-y = 1$  then the elements in  $SA$  in columns  $y$  and  $4m+3-x$  remain unaltered. Elements in rows  $x$  and  $4m+3-y$  remain unaltered.  $\square$

**Example 7.7** Following is an instance of the latin interchange  $I_{(3,6)}$  in  $S_3$  and its disjoint mate  $I'(3,6)$  for  $MBC(2,7)$  using Lemma 7.3:

$I_{(3,6)} =$ 

					14						1		
					8				4				
					4				3	2	14		
					1				8	3	2		

$I'_{(3,6)} =$ 

					1						14		
					4				8				
					14				4	3	2		
					8				3	2	1		

**Lemma 7.4** The coordinates of the cells  $\{(x,y)|x,y \in N\}$  in partial latin square  $S_4$  given in figure 7.6 are:

$x = \{3m + 6 - y, ..., y\}$  and  $y = \{\frac{3m+6}{2}, ..., 2m + 1\}$  for even order  $m$ ;  
 $x = \{3m + 6 - y, ..., y\}$  and  $y = \{\frac{3m+7}{2}, ..., 2m + 1\}$  for odd order  $m$ ;  
There exists a latin interchange  $I_{x,y}$  for every element  $\{(x,y;z)|x,y,z \in N\}$  in  $S_4$ .

**Proof.** Take a typical element  $(x,y;z)$  in  $S_4$  and construct the latin interchange  $I_{x,y}$  as follows: The three elements of  $I_{x,y}$  in the  $AR$  subsquare are

- If  $x \leq 3m+7-y$ , then
  - $(x-2, y; 3m+1)$
  - $(x, y-2; 3m+1)$
  - $(x, y; x+y-3 \bmod n)$
- Else
  - $(3m+5-y, y; 3m+1)$
  - $(x, 3m+5-x; 3m+1)$
  - $(x, y; m+3)$

End If

The two elements of  $I_{x,y}$  in the  $UpperBR$  subsquare are

**If**  $x < 3m+7-y$ , then

$(x-2, 2m+y-1; x+y-3 \bmod n)$

$(x, 2m+y-3; x+y-1 \bmod n)$

**Else If**  $x = 3m+7-y$ , then

$(x-2, 2m+y-2; x+y-3 \bmod n)$

$(x, 2m+y-4; x+y-1 \bmod n)$

**Else**

$(3m+5-y, 2m-2+y; m+3)$

$(x, 5m+3-x; x+y-1 \bmod n)$

**End If**

The two elements of  $I_{x,y}$  in the *LowerBR* subsquare are

**If**  $x < 3m+7-y$ , then

$(2m+x-3, y; x+y-1 \bmod n)$

$(2m+x-1, y-2; x+y-3 \bmod n)$

**Else If**  $x = 3m+7-y$ , then

$(2m+x-4, y; x+y-1 \bmod n)$

$(2m+x-2, y-2; x+y-3 \bmod n)$

**Else**

$(5m+3-y, y; x+y-1 \bmod n)$

$(2m+x-2, 3m+5-x; m+3)$

**End If**

For the elements of  $I_{x,y}$  in the *SA* subsquare, we have

**If**  $x < 3m+7-y$ , then

Pattern P1

$(2m+x-3, 2m+y-1; 3m+1)$

$(2m+x-1, 2m+y-3; 3m+1)$

$(2m+x-1, 2m+y-1; x+y-1 \bmod n)$

**Else If**  $x = 3m+7-y$ , then

$(2m+x-4, 2m+y-2; 3m+1)$

$(2m+x-2, 2m+y-4; 3m+1)$

$(2m+x-2, 2m+y-2; x+y-1 \bmod n)$

**Else**

Pattern P2

Let  $i = x+y-3m-7$

$(2m+x-4-i, 2m+y-2; 3m+1)$

$(2m+x-2, 2m+y-4-i; 3m+1)$

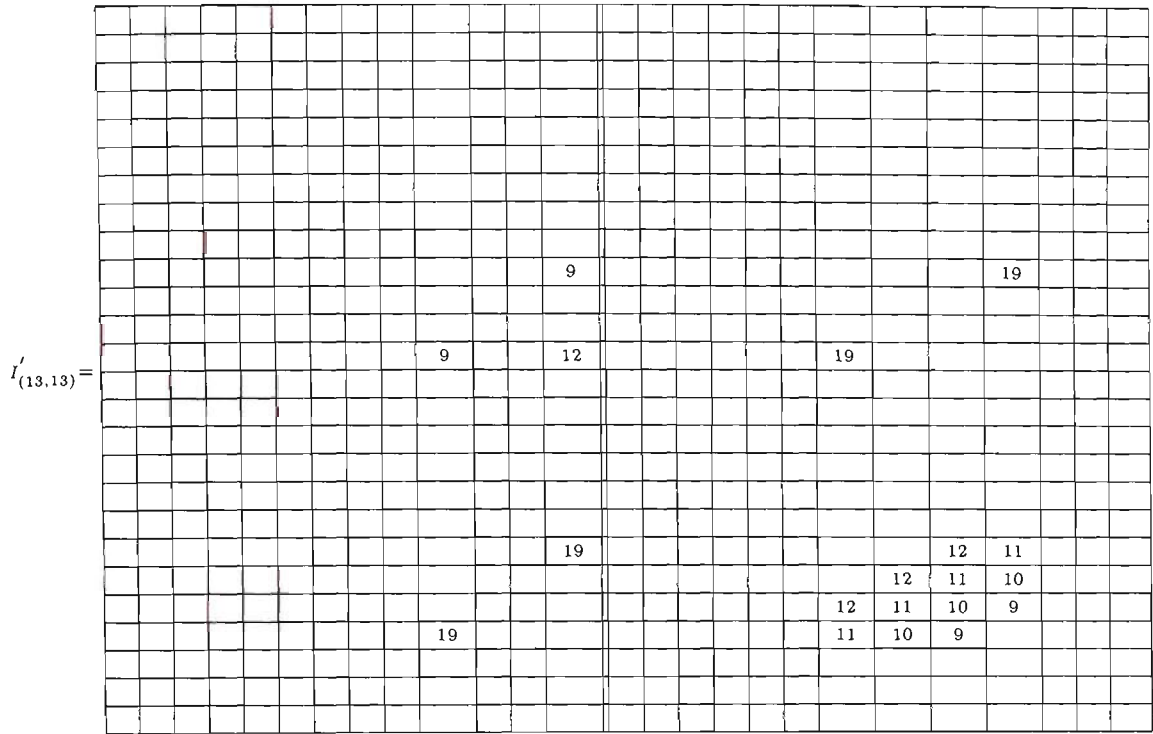
$(2m+x-3-i, 2m+y-3-i; m+4)$



[illegible]

**Example 7.9** Following is an instance of a latin interchange  $I_{(13,13)}$  and its disjoint mate  $I'(13,13)$  in  $S_4$  having pattern P2 in  $\text{MBC}(2,13)$  using Lemma 7.4:

[illegible]





$$\cup \{(i, j; i + j) : i = (n - 1)/2 + 1, \dots, n - 1 \text{ and } j = (n - 1)/2 - i, \dots, n - 1\}$$

where addition is reduced modulo  $n$ . They showed that sets  $P_1$  and  $P_2$  both satisfy condition 1 of the definition of critical set, i.e., say  $P_1$  is the only latin square of order  $n$  which has element  $k$  in position  $(i, j)$  for each  $(i, j, k)$ , and that  $P_1$  was in fact critical. Cooper, Donovan and Seberry [37] later verified that both  $P_1$  and  $P_2$  are critical sets, and that in fact  $P_1$  is minimal. They also showed that:

**Theorem 7.6**  *$P_1$  is a minimal critical set for a back-circulant latin square of even order  $n$ , and  $|P_1| = \frac{n^2}{4}$ .  $P_2$  is a critical set for a back-circulant latin square of odd order  $n$ , and  $|P_2| = \frac{n^2-1}{4}$ .*

Donovan and Cooper [54] later proved that:

**Lemma 7.7** *If  $L$  is any back-circulant latin square of order  $n$ , and  $r$  is some integer such that  $(n - 3)/2 \leq r \leq n - 2$ , then the set  $P = \{(i, j; i + j) : i = 0, \dots, r \text{ and } j = 0, \dots, r - i\} \cup \{(i, j; i + j) : i = r + 2, \dots, n - 1 \text{ and } j = r + 1 - i, \dots, n - 1\}$  is a critical set in  $L$ .*

Cooper, Donovan and Gower [39] proved the following theorem for the critical sets of the direct product of back-circulant latin squares containing subsquares of odd order:

**Theorem 7.8** *For  $n = 2m + 1$ ,  $n \geq 3$ , the partial latin square  $BC(2, n)$  given in Appendix B.4 and written as:*

$P_{2,n} = P \cup \{(i, j + n; k + n), (i + n, j, k + n), (i + n, j + n, k) | (i, j; k) \in BC_n\}$   
*is a critical set in the direct product of back-circulant latin squares  $BC(2, n) = BC_2 \times BC_n$*

In this chapter, the objective of studying latin interchanges was to construct a family of critical sets for modified-two back-circulant latin squares containing subsquares of odd order. We prove the following main theorem:

**Theorem 7.9** *For  $n = 2m + 1$ ,  $n \geq 3$ , the partial latin square  $MBC(2, n)$  given in Appendix B.4 and written as:*

$P_{2,n} = P \cup \{(i, j + n; k + n), (i + n, j; k + n), (i + n, j + n; n - k + 1) | (i, j; k) \in C_n\}$   
*is a critical set in the modified-two back-circulant latin squares  $MBC(2, n)$  for  $n \leq 25$ .*

**Proof.** The partial modified-two back-circulant latin square  $MBC(2, n) = MBC(2, 2m + 1)$  has the following form:

$$P_{2,n} = \begin{bmatrix} \text{par}AR & \text{par}BR \\ \text{par}BR & \text{par}SA \end{bmatrix}$$

where  $\text{par}AR$  is the partial square of the subsquare  $AR$ ,  $\text{par}BR$  is the partial square of the subsquare  $BR$  and  $\text{par}SA$  is the partial square of the subsquare  $SA$ .  $P_{2,n}$  is given in generalised form in Appendix B.4. It has been proved in Section 5.3 that  $P_{2,n}$  is uniquely completable. We have to show that  $P_{2,n}$  is also a critical set. To show that a uniquely completable set is critical, we only need to show that every element in it has a latin interchange. Now  $BR$  and  $SA$  subsquares are back-circulant latin squares of odd order  $n$ , hence  $\text{par}BR$  and  $\text{par}SA$  are the critical sets as shown in Lemma 7.7 above.

We now only have to show that every element in  $\text{par}AR$  has a latin interchange. In Section 7.2, partial subsquare  $\text{par}AR$  was further subdivided into 5 partial squares  $S_1, S_2, S_3, S_4$  and  $S_5$ . We have shown in Lemmas 7.1, 7.3 and 7.4 that every element in  $S_1, S_3$  and  $S_4$  has a latin interchange for all  $n$ . We have also shown in Lemma 7.2 that every element in  $S_2$  has a latin interchange in orders up to latin square of order 46. In Lemma 7.2 we have given formulae for general constructions of latin interchanges for all  $n$ . These formulae cover every element upto  $MBC(2, 23)$  but there are still a few cases in higher orders of  $MBC(2, n)$  for  $n > 23$  which are not covered by the formulae given in Lemma 7.2. In  $MBC(2, 25)$ , the elements in the cells (5,10), (6,10), (7,8), (7,9) and (8,8) are not covered by the given formulae. We have constructed the latin interchanges for each of these elements but not a formula. These latin interchanges are given in Appendix B.5. Hence every element in  $S_2$  has a latin interchange for  $n \leq 25$ . Since  $AR$  subsquare is a back-circulant latin square, it is symmetric. Hence all the elements in  $S_5$  are obtained by the reflection of the equivalent elements in  $S_1, S_2, S_3$  and  $S_4$ , i.e., element  $(i, j) = \text{element}(j, i)$  in  $AR$ . Thus every element in  $S_5$  has a latin interchange if and only if every element in  $S_1, S_2, S_3$  and  $S_4$  has a latin interchange. Hence every element in  $S_5$  has a latin interchange for  $n \leq 25$ .

Consequently every element in  $\text{par}AR$  has a latin interchange for  $n \leq 25$ . Thus  $P_{2,n}$  is a critical set of  $MBC(2, n)$  for  $n \leq 25$ . In orders for  $n > 25$ , while we believe that our uniquely completable set is still critical we have not been able to prove this.

**Conjecture 7.10** For  $n = 2m + 1$ ,  $n \geq 3$ , the partial latin square  $MBC(2, n)$  given in Appendix B.4 and written as:

$P_{2,n} = P \cup \{(i, j + n; k + n), (i + n, j; k + n), (i + n, j + n; n - k + 1)\} | (i, j; k) \in C_n$   
is a critical set in the modified-two back-circulant latin squares  $MBC(2, n)$  for all  $n$ .

**Example 7.10** Following uniquely completable set for an  $MBC(2,5)$  is a critical set.

1	2	3	4	5	6	7				40
2	3	4	5	1	7					
3	4	5	1	2						
4	5	1	2						8	
5	1	2						8	9	
6	7				5	4				
7					4					
				8					3	
			8	9				3	2	100

**Proof.** The three subsquares other than the top left subsquare are critical sets of back-circulant latin square of order 5 as proved by Cooper, Donovan and Seberry in [37]. We give latin interchanges for each entry in the top left-hand subsquare in Appendix B.5. These latin interchanges have been constructed using the formulae given by the lemmas in Section 7.2. □

## 7.4 Conclusion

In this chapter, general constructions of latin interchanges have been given for every element in  $S_1$ ,  $S_3$ , and  $S_4$  of partial modified-two back-circulant latin square of an  $MBC(2,n)$ , where  $n = 2m + 1$ . This shows that the indicated partial latin square in  $S_1$ ,  $S_3$ , and  $S_4$  is in critical set. It has been proved that the given partial  $MBC(2,n)$  is critical for all  $n < 25$ . For  $S_2$ , formulae have been given to construct latin interchanges containing different patterns but we have not yet proven that these formulae will cover all the cases for  $n \geq 25$ . More examples of such patterns are given in Appendix B.5. Hence the construction of formulae for the latin interchanges of the remaining unsolved cases in  $S_2$  for  $n \geq 25$  is still an open problem. There are also three open problems of whether the uniquely completable sets for  $MBC(2, 2m)$ ,  $BCFC(2,2m)$  and  $BCFC(2,2m+1)$ , given in Appendix B, are also critical sets.

## Part III

# Bhaskar Rao Designs

# Chapter 8

---

## Bhaskar Rao Designs of Block Size 5

Much of this chapter is based on the paper by Chaudhry, Greig and Seberry [27]. Many general results have been given by the second author and most of the numerical results have been contributed by the first author. More than 20 percent of the work in this paper is due to the first author. Section 8.6 is based on the paper by Chaudhry and Seberry [21]. More than 95 percent of the work in this paper is due to the first author. Overall about 25 percent of the work in this chapter is due to the author of this thesis.

### 8.1 Introduction

Bhaskar Rao designs (BRD) with elements  $0, \pm 1$  have been studied by a number of authors including Bhaskar Rao [16, 17], Chaudhry and Seberry [21], de Launey [41], de Launey and Sarvate [42], de Launey and Seberry [45, 46], de Launey, Sarvate and Seberry [44], Gibbons and Mathon [59], Lam and Seberry [78], Palmer and Seberry [90], Seberry [95, 96], Singh [99], Street [106], Street and Rodger [107], and Vyas [110]. BRDs have useful application in cryptographic functions and perfect hashing functions. BRDs with block size 3 were studied by Singh [99], Vyas [110], and Seberry [95, 96]. For block size 3, the necessary conditions are sufficient for all elementary Abelian groups and all groups of order less than or equal 8. BRDs with block size 4 for  $Z_2$  and all elementary Abelian groups were studied by de Launey and Seberry [45, 46]. See [43] for a recent survey which does not include the more recent work by Hard and Sarvate [67, 68, 69, 70], Greig, Hard and Sarvate [61], Greig, Hard, McCranie and Sarvate [62] and Deng, Greig and Osterg [47].

A *balanced incomplete block design* (BIBD) is an arrangement of  $v$  symbols in  $b$  blocks each containing  $(k < v)$  symbols, satisfying the following conditions :

- i). every symbol occurs at most once in a block,
- ii). every symbol occurs in exactly  $r$  blocks,

iii). every pair of treatments or symbols occur together in exactly  $\lambda$  blocks.

The incidence matrix  $N = (n_{ij})$  of a BIBD has entry one if symbol  $i$  is in block number  $j$ . The parameters  $(v, b, r, k, \lambda)$  of BIBD satisfy:

$$(i) \quad vr = bk \quad (ii) \quad \lambda(v-1) = r(k-1).$$

A *Bhaskar Rao design*,  $\text{BRD}(v, b, r, k, \lambda)$  is a matrix of order  $v \times b$  with  $(0, \pm 1)$  entries), satisfying the following conditions :

(i) the inner product of any pair of distinct rows is zero,

(ii) when its non-zero entries are replaced by  $+1$ 's, the resulting matrix becomes the incidence matrix of a  $\text{BIBD}(v, b, r, k, \lambda)$ .

In this chapter, the necessary conditions for the existence of Bhaskar Rao designs of block size 5 are:

$$(i) \quad \lambda(v-1) \equiv 0 \pmod{4}$$

$$(ii) \quad \lambda v(v-1) \equiv 0 \pmod{40}$$

$$(iii) \quad 2|\lambda.$$

Condition (i) follows from the necessary conditions of the BIBDs, condition (ii) follows from Theorem 8.2 and (iii) comes from Theorem 8.1. It is shown that these conditions are sufficient: for  $\lambda = 4$  if  $v > 215$ , with 10 smaller possible exceptions and one definite exception at  $v = 5$ ; for  $\lambda = 10$  if  $v > 445$ , with 11 smaller possible exceptions, and one definite exception at  $v = 5$ ; and for  $\lambda = 20$ , with the possible exception of  $v = 32$ ; we also give a few results for other values of  $\lambda$ . We use the notations  $\text{BIBD}(v, k, \lambda)$  for  $\text{BIBD}(v, b, r, k, \lambda)$  and  $\text{BRD}(v, k, \lambda)$  for  $\text{BRD}(v, b, r, k, \lambda)$ , omitting  $b$  and  $r$  as they are dependent on  $v, k, \lambda$ . In the examples, we write ‘ $-$ ’ for ‘ $-1$ ’. The following example gives a BRD of block size 5.

**Example 8.1** There exists a  $\text{BRD}(6, 5, 4)$  constructed from  $\text{BIBD}(6, 5, 4)$ .

$\text{BIBD}(6, 5, 4)$

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$\text{BRD}(6, 5, 4)$

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & - & - & 1 \\ 1 & 1 & 0 & 1 & - & - \\ 1 & - & 1 & 0 & 1 & - \\ 1 & - & - & 1 & 0 & 1 \\ 1 & 1 & - & - & 1 & 0 \end{bmatrix}$$

In fact, a  $\text{BRD}(k+1, k, k-1)$  exists whenever  $k$  is an odd prime power [48].

**Theorem 8.1 (Bhaskar Rao [17])** *A necessary condition for the existence of BRD is that  $\lambda$  must be even.*

A *pairwise balanced design*,  $\text{PBD}(v, K, \lambda)$ , of order  $v$  with block sizes from  $K$ , where  $K$  a set of positive integers, is a pair  $(\mathcal{V}, \mathcal{B})$  where  $\mathcal{V}$  is a finite set of cardinality  $v$  and  $\mathcal{B}$  is a family of blocks of  $\mathcal{V}$  which satisfies the properties:

- i). If  $B \in \mathcal{B}$ , then  $|B| \in K$
- ii). Every pair of distinct elements of  $\mathcal{V}$  occurs in exactly  $\lambda$  blocks of  $\mathcal{B}$ .

If  $K = \{k\}$ , then the PBD is a BIBD.

A *group divisible design*,  $\text{GDD}_\lambda(K, G)$ , of order  $v$ , where  $K$  a set of positive integers, is a triple  $(\mathcal{V}, \mathcal{G}, \mathcal{B})$  where  $\mathcal{V}$  is a finite set of cardinality  $v$ ,  $\mathcal{G}$  is a partition of  $\mathcal{V}$  into groups  $G_1, G_2, \dots, G_s$ , and the vector of group sizes  $G = (|G_1|, |G_2|, \dots, |G_s|)$  (often written in exponential notation) is known as the type of the GDD, and  $\mathcal{B}$  is a family of blocks of  $\mathcal{V}$  which satisfies the properties :

- i). If  $B \in \mathcal{B}$ , then  $|B| \in K$ ;
- ii). Every pair of distinct elements of  $\mathcal{V}$  occurs in exactly  $\lambda$  blocks or one group, but not both.

When the index is one, (i.e.,  $\lambda = 1$ ), the subscript  $\lambda$  is often omitted from a  $\text{GDD}_\lambda(K, G)$ . A GDD with group type  $G = 1^v$  is a PBD.

A *transversal design*,  $\text{TD}_\lambda(k, n)$ , of order  $n$ , block size  $k$ , and index  $\lambda$  is a  $\text{GDD}_\lambda(k, n^k)$ .

A *resolvable design* is a design whose block set admits a partition into parallel classes, or resolution sets, each of which contains every point exactly once. Resolvable designs are indicated by the prefix R.

A set of  $k-1$  mutually orthogonal latin squares, (or MOLS), of order  $n$ , a  $\text{RTD}(k, n)$  and a  $\text{TD}(k+1, n)$  are three equivalent combinatorial objects. A  $\text{RTD}(k, k)$  exists whenever  $k$  is a prime power. Note that if we have a  $\text{RTD}(k, n)$ , we also have a  $\text{RTD}(k', n)$  for  $k' < k$ ; (just throw away  $k - k'$  of the MOLS).

A  $\text{BRGDD}_\lambda(K, G)$  of group type  $G = (|G_1|, |G_2|, \dots, |G_g|)$ , is a matrix of order  $v \times b$  with  $(0, \pm 1)$  entries, satisfying the following conditions :

- i). the inner product of any pair of distinct rows is zero,

- ii). when its non-zero entries are replaced by +1's, the resulting matrix becomes the incidence matrix of a  $\text{GDD}_\lambda(K, G)$  of group type  $G$ .

BRGDDs with a uniform group size, signed over more general groups, were studied by Palmer under the name partial generalized Bhaskar Rao designs, in [89].

**Theorem 8.2 (Seberry [95])** *A Bhaskar Rao design  $\text{BRD}(v, k, \lambda)$  can only exist if the equation*

$$(i) \quad x_3 + 3x_5 + 6x_7 + \dots + ((k^2 - 1)/8)x_k = b(k - 1)/8 \text{ for } k \text{ odd,}$$

$$(ii) \quad x_0 + 3x_4 + 8x_6 + \dots + ((k^2 - 4)/4)x_k = b(k - 4)/4 \text{ for } k \text{ even,}$$

*has integral solutions. In particular, for  $k \equiv 3 \pmod{4}$ , a Bhaskar Rao design can only exist if  $4|b$ ; for  $k \equiv 0, 1, 4 \pmod{8}$  no restriction is obtained; for  $k \equiv 2, 5, 6 \pmod{8}$  we must have  $2|b$ . Thus, for  $k = 5$ ,  $b/2$  must be an integer.*

**Theorem 8.3** *Suppose that the BIBD underlying a  $\text{BRD}(v, k, \lambda)$  has  $\lambda$  identical blocks. If  $k \geq 3$ , then it is necessary that  $\lambda \geq k$  and  $4|\lambda$ .*

**Proof:** Suppose the repeated block contains  $(a, b, c, \dots)$ , and suppose w.l.o.g. that all  $a$ 's get +, then since half the  $(a, b)$  pairs have mixed sign, half the  $b$ 's are + and half −; (similarly the  $c$ 's). Also half the  $(b, c)$  pairs have mixed sign. Now suppose there are  $x$   $(b, c)$  pairs of type  $(+1, +1)$ ; then there are  $\lambda/2 - x$  mixed pairs with  $b$  positive, and the same number with  $c$  positive, hence  $\lambda/2 = 2(\lambda/2 - x)$ , which implies  $4|\lambda$ . Finally, consider the  $k \times \lambda$  submatrix,  $M$ , within the signed incidence matrix that corresponds to these blocks. We have  $MM^T = \lambda I$ , which has rank  $k$ , and so therefore does  $M$ , and we cannot have a dimension smaller than the rank.  $\square$

**Theorem 8.4** *A  $\text{BRD}(k, k, 4t)$  exists whenever a Hadamard design of order  $4t$  exists, with  $4t \geq k$ .*

**Proof:** Take the first  $k$  rows of the Hadamard design as the BRD.  $\square$

**Theorem 8.5 (de Launey [41])** *For all  $k, t > 0$ , there is an integer  $M$  such that if  $2t(v-1)/(k-1)$  and  $2tv(v-1)/k(k-1)$  are integers and  $v > M$ , then a  $\text{BRD}(v, k, 2t; \mathbb{Z}_2)$  exists.*



**Theorem 8.6 (Lam and Seberry [78])** *Suppose there exists a  $BRD(v, k, \lambda)$  and  $BRD(u, k, \mu)$ . Further suppose there are  $k-1$  mutually orthogonal latin squares of order  $u$ . Then there exists a  $BRD(uv, k, \mu\lambda)$ .*

**Theorem 8.7 (de Launey and Seberry [41])** *Suppose there exists a  $BRD(u, k, \lambda)$  with a subdesign on  $w$  points ( $w = 0, 1$  are allowed), a  $BRD(v, k, \lambda)$  and  $k-2$  mutually orthogonal latin squares of order  $u - w$ . Then there exists a  $BRD(v(u - w) + w, k, \lambda)$  with sub-designs on  $u, v$  and  $w$  points.*

**Theorem 8.8 (Seberry [95])** *Suppose there exists a  $BRD(v, b, r, k; 4t)$ ,  $4t$  is the order of a Hadamard matrix and there exist  $k-1$  mutually orthogonal latin squares of order  $k$ . Then there exists a  $BRD(kv, 4tv + k^2b, kr + 4t, k, 4t)$ .*

**Corollary 8.9 (Seberry [95])** *Suppose there exists a  $BRD(v, b, r, k; 4)$ , then there exists a  $BRD(kv, 4v + k^2b, kr + 4, k, 4)$  whenever  $k$  is a prime power.*

## 8.2 $BRD(v, 3, \lambda)$ and $BRD(v, 4, \lambda)$

The main theorem by Seberry for block size 3 is :

**Theorem 8.10 (Seberry [95])** *The conditions  $\lambda \equiv 0 \pmod{2}$ ,  $\lambda v(v - 1) \equiv 0 \pmod{24}$  are the necessary and sufficient conditions for the existence of  $BRD(v, 3, \lambda)$  for  $v \geq 3$ .*

Singh [99] showed that if  $v \leq 40$ ;  $vr = bk$ ,  $\lambda(v - 1) = r(k - 1)$  and  $4|b$  are necessary and sufficient conditions for the existence of Bhaskar Rao designs  $(v, b, r, 3, 2)$  or  $BRD(v, 3, 2; Z_2)$ . Vyas [110] also showed that if  $v \leq 40$ ;  $vr = bk$ ,  $\lambda(v - 1) = r(k - 1)$  and  $4|b$  are necessary and sufficient conditions for the existence of Bhaskar Rao designs  $(v, b, r, 3, 4)$  or  $BRD(v, 3, 4; Z_2)$ .

Now we look at work done by de Launey and Seberry [45, 46] on Bhaskar Rao designs for block size 4. They proved the following theorems.

**Theorem 8.11** *A  $BRD(v, 4, 2)$  exists for  $v \equiv 1 \pmod{6}$  is a prime power. A  $BRD(v, 4, 2)$  can exist only if  $v \equiv 1, 4 \pmod{6}$ .*

**Remark 8.2.1** There is no  $BRD(4, 4, 2)$  [46] or  $BRD(10, 4, 2)$  [42].

**Theorem 8.12** *Let  $p \geq 5$ , odd, be a prime or prime power. Then there is a  $BRD(p, \frac{1}{2}p(p-1), 2(p-1), 4, 6)$ .*

**Theorem 8.13** *A  $BRD(v, 4, 12)$  exists for all  $v \geq 4$ .*

**Theorem 8.14** *A  $BRD(v, 4, 2t)$  exists for  $v > 4$  and  $t > 1$  if and only if  $t(v-1) \equiv 0 \pmod{3}$  except possibly when  $t = 3, 5, 7$  and  $v = 28, 34, 39$  or when  $t = 3$  and  $v = 39$ . A  $BRD(4, 4, 2t)$  exists for  $t > 0$  if and only if  $t$  is even.*

Improving on the work of de Launey and Seberry [46], Greig, Hard, McCranie and Sarvate [62] and Deng, Greig and Osterg [47] have shown:

**Theorem 8.15** *The necessary conditions for a  $BRD(v, 4, \lambda)$  with  $v > 4$ , namely*

$$(i) \quad \lambda(v-1) \equiv 0 \pmod{3}$$

$$(ii) \quad \lambda v(v-1) \equiv 0 \pmod{12}$$

$$(iii) \quad \lambda \equiv 0 \pmod{2}$$

*are sufficient, except for  $v \equiv 4 \pmod{6}$  and  $\lambda = 2$ , where there is one definite exception (for  $v = 10$ ) and 28 other possible exceptions in the range  $22 \leq v \leq 430$ .*

## 8.3 General Constructions

We start by adapting some classical constructions to Bhaskar Rao type designs. The most useful is a generalization of Wilson's Fundamental Construction (WFC):

**Theorem 8.16** *Suppose we have a master  $GDD_\lambda(K', G)$  with group type given by  $G = (|G_1|, \dots, |G_s|)$ . Suppose  $w(x)$  is a positive weighting function defined for each point of the master design. For each block  $B = \{b_1, \dots, b_{k'}\}$ , assume we have an ingredient  $GDD_\mu(K, W(B))$  with group type  $W(B) = (|w(b_1)|, \dots, |w(b_{k'})|)$ . Then there is a  $GDD_{\lambda\mu}(K, W(G))$  with group type*

$$W(G) = \left( \sum_{x \in G_1} w(x), \dots, \sum_{x \in G_s} w(x) \right).$$

*Furthermore, if either the master design, or all the ingredient designs are BRGDDs, (or both), then so is the resultant design.*

**Proof:** The proof of the basic WFC is available in, for example, [15, IX.3.2]; in our variant, again the resultant's  $K$  is not directly dependent on the master's  $K'$ . For the BRGDD version, suppose we are looking at a master block containing  $b_i$  with a sign of  $g_1(b_i)$ , and in a block of the appropriate ingredient design, we have  $w_j(b_i)$  with a sign of  $g_2(w_j(b_i))$ , then in the resultant design we give the point a sign of  $g_1(b_i) * g_2(w_j(b_i))$ .

**Remark 8.3.1** This theorem can be generalized to BRGDDs over groups other than  $Z_2$ .

We next look at filling in the groups of the BRGDD. The first construction is usually applied to each group in turn.

**Theorem 8.17** *Suppose we have a  $BRGDD_\lambda(K, G)$  with group type  $G$ , where  $G = (G_1, G_2, \dots, G_s)$ . Let  $H = (H_1, H_2, \dots, H_t)$ , and  $|G_1| = \sum |H_j|$ ; if we also have a  $BRGDD_\lambda(K, H)$ , then we have a  $BRGDD_\lambda(K, F)$  with group type*

$$F = (|H_1|, |H_2|, \dots, |H_t|, |G_2|, \dots, |G_s|).$$

**Theorem 8.18** *Let  $\omega \geq 0$ . Suppose we have a  $BRGDD_\lambda(K, G)$  with group type  $G = (|G_1|, |G_2|, \dots, |G_s|)$ , and for the  $i$ -th group (with  $i > 1$ ), we have a  $BRGDD_\lambda(K, H_i)$  with group type  $H_i = (\omega, |H_{i1}|, |H_{i2}|, \dots, |H_{it_i}|)$ , and  $|G_i| = \sum_j |H_{ij}|$ ; then we have a  $BRGDD_\lambda(K, F)$  with group type*

$$F = (\omega + |G_1|, |H_1|, |H_2|, \dots, |H_s|).$$

We can derive known results as corollaries of these constructions. We give an example:

**Theorem 8.19 (Lam and Seberry [78])** *Let  $w \in \{0, 1\}$ . Suppose there exists a  $BRD(v, k, \lambda)$  and  $BRD(u + w, k, \lambda)$ , further suppose there is a  $TD(k, u)$ ; then there exists a  $BRD(uv + w, k, \lambda)$ .*

**Proof.** Take the  $BRD(v, k, \lambda)$  as the master in the WFC, and give each point a weight of  $u$ . The TD provides the ingredient, and generates a  $BRGDD_\lambda(k, u^v)$ . Then use Theorem 8.18 with the  $BRD(u + w, k, \lambda)$  providing a group type of  $1^{u+w}$  to get the result.  $\square$

**Remark 8.3.2** Actually, the original construction of Theorem 8.19 given as Theorem 8.6 was weaker than our current version; they required a  $TD(k + 1, u)$ , only allowed  $w = 0$ , and got a larger final index.

Another derivable general construction is the following singular direct product construction:

**Theorem 8.20 (de Launey and Seberry [45])** *Suppose there exists a  $BRD(u, k, \lambda)$  with a subdesign on  $w$  points, a  $BRD(v, k, \lambda)$  and  $k - 2$  mutually orthogonal latin squares of order  $u - w$ . Then there exists a  $BRD(v(u - w) + w, k, \lambda)$  with sub-designs on  $u, v$  and  $w$  points, ( $w = 0, 1$  are allowed).*

**Proof.** Take the  $BRD(v, k, \lambda)$  as the master design (with type  $1^v$ ) and give points a weight of  $u - w$  in WFC, then use Theorem 8.18 to get the result.  $\square$

**Remark 8.3.3** Actually, we do not need the  $BRD(u, k, \lambda)$  to have a subdesign on  $w$  points, what we really need is a  $BRGDD_\lambda(k, 1^{u-w}w^1)$  to deal with all but one of the groups, and a type  $1^u$ , (i.e., the BRD itself), for the final group. To give a concrete example of the distinction, we note that if  $n \in \{4, 6, 10, 12, 15, 18\}$ , or  $n > 103$  then there is a  $BRGDD_4(5, 1^{10n}3^1)$ . See [79, lemmas 112–118]; these BRGDDs are all derived from  $GDD(5, 2^{10n}6^1)$ . The cases  $n = 4$  and  $6$  are constructed directly. The case  $n = 18$  is generated by a  $GDD(5, 10^9)$  with five parallel classes constructed by Abel; all the others are produced recursively. So we could take  $u = 40, w = 3, \lambda = 20, v = 8, k = 5$  say; the point here is that clearly we can't actually have any  $BRD(w, k, \lambda)$  since here  $w < k$ .

**Theorem 8.21 (Seberry [95])** *Suppose there exists a  $BRD(v, b, r, k, 4t)$ ,  $k \leq 4t$  and  $4t$  is the order of a Hadamard matrix and there exist  $k - 2$  mutually orthogonal latin squares of order  $k$ . Then there exists a  $BRD(kv, 4tv + k^2b, kr + 4t, k, 4t)$ .*

**Proof.** Use the BRD as the master design, giving points a weight of  $k$ , and the  $TD(k, k)$  as the ingredient, then fill the groups with a  $BRD(k, k, 4t)$  formed from the first  $k$  rows of the Hadamard matrix.  $\square$

**Remark 8.3.4** The original statement of the theorem omitted the condition  $k \leq 4t$ , although its use is apparent in the original proof, which is essentially the same construction as our proof. The original theorem also called for  $k - 1$  MOLS; since the order is  $k$ , this is a distinction which does not matter for any value of  $k$ , although we have stated  $k - 2$  to match the TD we have used.

Palmer [89] also has some constructions, which when restricted to signing over  $Z_2$ , are special cases of Theorem 8.16.

## 8.4 Signing Known Designs

The obvious way to construct BRDs is to take the underlying BIBD, and change the signs of the elements of its incidence matrix in some suitable way.

**Lemma 8.22** *Suppose we have a partially signed BIBD( $v, k, \lambda$ ) incidence matrix of the form  $\begin{bmatrix} A \\ B \end{bmatrix}$  such that the rows of  $A$  are mutually orthogonal and the rows of  $B$  are also mutually orthogonal. Then  $C = \begin{bmatrix} A & A \\ B & -B \end{bmatrix}$  is a BRD( $v, k, 2\lambda$ ).*

**Example 8.2** A BRD(11, 5, 4) exists.

We note that if we change all the signs of any point, or of any block, then we will still preserve the orthogonality (or lack of it) for any pair of points. We take the unique BIBD(11, 5, 2) and commence to sign it, requiring the first element in each row and column to be +1. The signing of the first 5 rows is unique, and we can compatibly sign any one of the remaining rows, (row 6 in the example below), also in a unique way, however no pair of the compatibly signed last six rows is orthogonal, so make row 6 orthogonal to the first 5, then start again. We have signed the last five rows in an orthogonal fashion, with  $a, b \in \{\pm 1\}$ ; we can also easily check that it is not possible to sign row 6 in a way that makes the last six rows orthogonal. That we should encounter problems is to be expected, since, by Theorem 8.42, we know that we cannot sign a BIBD(11, 5, 2). However since we have the first 6 rows mutually orthogonal and similarly the last five so we may apply Lemma 8.22 and get a BRD(11, 5, 4).

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & - & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & - & 0 & 0 & - & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & - & 0 & 0 & - & 0 & - & 0 & 1 \\ 1 & 0 & 0 & 0 & - & 0 & 0 & - & 0 & - & - \\ 0 & 1 & - & 0 & 0 & 0 & 0 & 1 & - & 0 & - \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & - & 1 \\ 0 & 0 & 1 & 0 & a & b & \bar{a} & 0 & 0 & 0 & \bar{b} \\ 0 & 0 & 1 & \bar{a} & 0 & 0 & a & 1 & 0 & \bar{a} & 0 \\ 0 & 0 & 0 & 1 & ab & - & 0 & a & a\bar{b} & 0 & 0 \end{bmatrix}$$

**Example 8.3** A construction of a  $RTD(8, 8)$  from the multiplication table of a  $GF(2^3)$  where the table is indexed by zero and the powers of a root of the primitive equation  $x^3 + x + 1 = 0$ .

	0	$x^0$	$x^1$	$x^2$	$x^3$	$x^4$	$x^5$	$x^6$
0	000	000	000	000	000	000	000	000
$x^0$	000	001	010	100	011	110	111	101
$x^1$	000	010	100	011	110	111	101	001
$x^2$	000	100	011	110	111	101	001	010
$x^3$	000	011	110	111	101	001	010	100
$x^4$	000	110	111	101	001	010	100	011
$x^5$	000	111	101	001	010	100	011	110
$x^6$	000	101	001	010	100	011	110	111

To form the design append the row label to each element. Take each column as a base block, and develop over  $Z_2 \times Z_2 \times Z_2$  for the RTD. Each base block generates a parallel class.

**Theorem 8.23** *Let  $m$  and  $n$  be non-negative integers and  $q$  a prime with  $k \leq q^{m+n+1}$ . Then a  $BRGDD_{q^{m+1}}(k, (q^n)^k)$  signed over  $GF(q)$  exists.*

**Proof:** We proceed as in Example 8.3 to form the  $RTD(q^{m+n+1}, q^{m+n+1})$ , but we retain only the first  $k$  rows (thus forming a  $RTD(k, q^{m+n+1})$ ). Now we use the first element of the  $m+n+1$ -tuple to “sign” the tuple with an element of  $Z_q$ , and we then ignore the next  $m$  elements, and only develop each of the last  $n$  elements over  $Z_q$ . (Conventionally, for  $Z_2$ , we can replace the first element 0 by a plus sign, and the first element 1 by a minus sign).  $\square$

**Corollary 8.24** *The following designs exist:*

- i). A  $BRGDD_4(8, 2^8)$  and a  $BRGDD_4(5, 2^5)$ .
- ii). A  $BRGDD_2(8, 4^8)$  and a  $BRGDD_2(5, 4^5)$ .
- iii). A  $BRGDD_2(16, 8^8)$  and a  $BRGDD_2(5, 8^5)$ .

**Theorem 8.25** *Suppose we have a  $GDD_\lambda(k, G)$  with groups  $G_1, \dots, G_s$ , such that  $|G_i|$  is even for all  $i$ ; then we have a  $BRGDD_{4\lambda}(k, H)$  with group list  $H = \{|G_i|/2 : 1 \leq i \leq s\}$ .*

**Proof:** Replace the points  $a_1, \dots, a_{2t}$  in  $G_i$  by  $b_1, \dots, b_t$ , substituting  $b_j$  in blocks containing  $a_{2j}$ , and substituting  $-b_j$  in blocks containing  $a_{2j-1}$ .  $\square$

**Remark 8.4.1** This theorem generalizes to “signings” over  $GF(q)$  under the condition that  $q$  divides  $|G_i|$ , and with the initial GDD index of  $\lambda$  being multiplied by  $q^2$  in the BRGDD.

**Remark 8.4.2** An extension of the above, is that if the design was given by base blocks that are developed over the group, assuming now a uniform group size of  $|G|$ , then we may divide the resulting index by  $|G|$ .

This is a very useful construction, because it allows us to get designs with almost no effort from the literature.

**Example 8.4.3** We take as an example a  $GDD(5, 2^{61})$  developed over  $Z_2 \times Z_{61}$  given in [114], and replace the  $Z_2$  elements (0 by +1, 1 by -1) to get the base blocks  $(0, 1, 4, 25, -11)$  and  $(0, 8, 23, -25, -27)$ ; multiply these by 1, 13 and 47, (i.e., the cube roots in  $Z_{61}$ ), to get 6 base blocks which when developed over  $Z_{61}$  yield a  $BRD(61, 5, 2)$ .

**Theorem 8.26** *If  $v \in \{41, 61, 81\}$ , then there is a  $BRD(v, 5, 2)$ .*

**Proof:** See [114, Lemmas 2.1 and 2.7]; their constructions of a  $GDD(5, 2^v)$  are by base blocks developed over  $Z_2 \times Z_v$ .  $\square$

**Theorem 8.27** *If  $p \equiv 1 \pmod{10}$  is a prime and  $41 \leq p \leq 1151$ , then there is a  $GDD(5)$  of type  $4^p$  constructed from (block-disjoint) base blocks over  $GF(4) \times Z_p$ .*

**Proof:** See [9, Table 2.1, Theorem 2.2].  $\square$

**Corollary 8.28** *If  $p \equiv 1 \pmod{10}$  is a prime and  $41 \leq p \leq 1151$ , then there is a  $BRGDD_2(5, 2^p)$  of type  $2^p$ .*

**Proof:** Converting  $Z_4$  into a signed  $GF(2)$  increases the index to the 4, but we may halve this since we had developed over  $Z_4$  and now only develop over  $GF(2)$  subgroups.  $\square$

We begin by presenting a standard construction for  $AG(2, q)$  where  $q$  is a prime power, (see e.g., [66, Theorem 2.1]. Let  $x$  be a primitive element for  $GF(q)$ .

$$\mathcal{P} = GF(q) \times GF(q)$$

$$G = \{(0, 0)(0, x^0)(0, x^1) \dots (0, x^{q-2})\} \bmod (q, -)$$

$$C = \{(0, 0)(x^0, 0) \dots (x^{q-2}, 0)\} \bmod (-, q)$$

$$B_\alpha = \{(0, 0)(x^0, x^\alpha)(x^1, x^{\alpha+1}) \dots (x^{q-2}, x^{\alpha+q-2})\} \bmod (-, q)$$

$$B = \text{dev}(G) \cup \text{dev}(C) \cup \bigcup_{\alpha=0}^{q-2} \text{dev}(B_\alpha)$$

We now adapt a construction of Wilson's, quoted in [65, Theorem 15.7.4], to produce BRGDDs.

**Theorem 8.4.4** *A BRGDD<sub>2</sub>( $q, t^{q+1}$ ) exists whenever  $q = 2t+1$  is an odd prime power.*

**Proof.** We perform a signed replacement in the above  $AG(2, q)$ , replacing the second element  $x^j$  by  $x^{j-t}$  if  $t \leq j < 2t$  and giving the point a negative sign, and omitting all points whose second element is zero, and giving the remaining points a positive sign. We also discard all the type  $|G|$  blocks, (which contain all the doubleton of the new points), and these now define the groups, so (finite) points are in the same group iff they have the same first element. Let us now examine when the  $(++)$  pair  $(x^a, x^j)$  with  $(x^b, x^k)$  occurs, (assuming  $j \neq k$ , and  $j, k < t$ ). If this happens in  $B_\alpha + d$ , then we have

$$x^j = x^{\alpha+a} + d$$

$$x^k = x^{\alpha+b} + d$$

so we have

$$x^j - x^k = x^\alpha(x^a - x^b)$$

$$d = x^j - x^{\alpha+a}$$

Now, by a similar calculation, (noting that  $x^{j+t} = -x^j$  in  $GF(q)$ ), the  $(--)$  pair will occur occur in  $B_{\alpha+t} + x^t d$ , and if  $d = -x^{\alpha+c}$  then the missing element from the deleted zeros, will be  $(x^c, 0)$  in both cases. Similarly, we can (using new  $\alpha$  and  $d$ ) find that the  $(+-)$  pair occurs when

$$x^j + x^k = x^\alpha(x^a - x^b)$$

$$d = x^j - x^{\alpha+a}$$



and again, the  $(-+)$  pair occurs in  $B_{\alpha+t} + x^t d$ , and again the two blocks have a common missing element. Next, for the pairs with  $(0, x^j)$  we have  $x^k - x^j = x^{\alpha+b}$ , for the  $(++)$  pair, and again the  $(--)$  pair occurs in  $B_{\alpha+t} + x^t d$ , and similarly the mixed sign pairs with  $x^k + x^j = x^{\alpha+b}$ , for the  $(-+)$  pair. If we have  $j = k$ , then it can be checked that we have no same sign solutions in the  $B_\alpha$  type blocks, although these can be obtained from  $C + x^j$ ,  $C + x^{j+t}$ .

To summarise what we have done so far; we removed the parallel class  $\text{dev}(G)$  to provide groups, then did a 2 to 1 collapsing of points within a group, (if we ignore the signs for now), which normally inflates the index by 4, but we have shown that the cyclotomic way we collapsed the points allows us to identify doubles of each of the blocks in the design, so we only need increase the index by 2, not 4. More generally, Wilson showed if  $q = ef + 1$  is a prime power, we can get a  $\text{GDD}_e(k, f^{q+1})$  by collapsing along cyclotomic lines. Furthermore, the signs are properly balanced in each half, so we have a BRGDD, but, so far, we have mixed block sizes, since all the  $B_\alpha$  type blocks have lost a point, and the  $C$  type blocks remain intact (but  $C$  did lose a whole block). Each original point occurred once in  $\text{dev}(B_\alpha)$  for each  $\alpha$ , so it has each new point twice, once with each sign, so that we can add a group of  $t$  (positive) infinite elements to the design, adding  $\infty_\alpha$  to  $\text{dev}(B_\alpha)$  to get our BRGDD.  $\square$

Since this is a symmetric design, (i.e.,  $v = b$ ), it is also a weighing matrix, which yields the following corollary.

**Corollary 8.4.5** *If  $p$  is an odd prime power, a  $W((p^2 - 1)/2, p)$  exists.*

**Corollary 8.4.6** *A  $\text{BRGDD}_2(5, 2^6)$  exists.*

**Example 8.4** Consider how we might construct a  $\text{BRD}(6, 3, 4)$  using some SDS. This design has 20 blocks, which suggests we use  $Z_{v-1}$  since  $Z_v$  would require some short orbits, which complicates things somewhat. (Here it looks like it just complicates things, but it can make the construction impossible, bearing Theorem 8.3 in mind.) Next, we see that we need 4 mixed differences amongst the 8 differences in the 5 finite elements, and also two finite elements of each sign in the base blocks containing  $\infty$ , (signing  $\infty$  positively). If we take 2 copies of the one-rotational  $2 - (6, 3, 2)$  SDS:  $(\infty, 0, 1)(0, 2, 4)$ , then we only have two options here; either we mix signs on the two infinite blocks, or we do not. In this case both yield solutions:

$$\begin{array}{cccccc} (\infty, 0, 1) & (\infty, -0, -1) & (0, 2, -4) & (0, 2, -4) & (\text{mod } 5) \\ (\infty, 0, -1) & (\infty, 0, -1) & (0, -2, 4) & (0, 2, 4) & (\text{mod } 5) \end{array}$$

Later, we will construct several examples of signing SDSs to construct BRDs.

8.5  $BRD(v, 5, \lambda)$

**Theorem 8.5.1** *For the existence of a  $BIBD(v, k, \lambda)$ , the necessary conditions are :*

- i).  $\lambda(v - 1) \equiv 0 \pmod k$
- ii).  $\lambda v(v - 1) \equiv 0 \pmod {k(k - 1)}$

*In the case that  $k = 5$ , these conditions are sufficient with the exception of the non-existent  $BIBD(15, 5, 2)$ .*

*In the case that  $k = 6$  and  $\lambda > 1$ , these conditions are sufficient with the exception of the non-existent  $BIBD(21, 6, 2)$ . In the case that  $k = 6$  and  $\lambda = 1$ , these conditions are sufficient with with possible exception of  $BIBD(v, 6, 1)$  for the 36 values of  $v$  in Table 8.5.1 below, (and where the values there with  $v \leq 46$  are definite exceptions).*

**Proof:** For  $k = 5$  and  $k = 6$  with  $\lambda > 1$ , see Hanani [66]. For  $BIBD(v, 6, 1)$ , the list is an updated version of [8, I.2.5] provided by Abel [5]. □

Table 8.5.1

Possible $BIBD(v, 6, 1)$ exceptions									
(16)	(21)	(36)	(46)	51	61	81	166	226	231
256	261	286	291	316	321	346	351	376	406
411	436	441	471	496	501	526	561	591	616
646	651	676	771	796	801				

**Theorem 8.5.2** *For the existence of a  $BRD(v, 5, \lambda)$ , the necessary conditions are:*

- i).  $\lambda(v - 1) \equiv 0 \pmod 4$
- ii).  $\lambda v(v - 1) \equiv 0 \pmod {20}$
- iii).  $\lambda \equiv 0 \pmod 2$

*Condition (iii) can be replaced by  $\lambda v(v - 1) \equiv 0 \pmod {40}$ . Furthermore, if  $v = 5$  we must have  $\lambda \equiv 0 \pmod 4$ , with  $\lambda > 4$ .*

For  $BRD(v, 5, \lambda)$ , these conditions imply the following:

**Table 8.5.2**

$\lambda$	condition on $v$
2	$v \equiv 1$ or $5 \pmod{20}$ , with $v > 5$
4	$v \equiv 0$ or $1 \pmod{5}$
10	$v \equiv 1 \pmod{4}$ , with $v > 5$
20	any $v \geq 5$

**Lemma 8.5.3** *A  $BRD(5, 5, \lambda)$  exists iff  $\lambda \equiv 0 \pmod{4}$  with  $\lambda \geq 8$ .*

**Proof.** This essentially is a corollary of Theorems 8.3 and 8.4, noting that Hadamard matrices of order 8 and 12 exist.  $\square$

**Theorem 8.29** *If a  $BIBD(v, 5, \lambda)$  exists, then a  $BRD(v, 5, 4t\lambda)$  exists for any  $t > 1$ .*

**Proof:** Take the BIBD as the master design in WFC, give each point a weight of 1, and use the BRD of Lemma 8.5.3 as the ingredient design. In this case, this construction just amounts to replacing each block by the  $BRD(5, 5, 4t)$  whose point set equals the points of the block.  $\square$

**Remark 8.5.4** Unfortunately, this general construction gives designs with  $\lambda$ 's that are too large; in fact, the only case of use is for  $v = 35$ , where we failed to find either a  $BRD(35, 5, 4)$  or a  $BRD(35, 5, 8)$ , but via the  $BIBD(35, 5, 2)$ , have  $BRD(35, 5, \lambda)$  for  $\lambda = 16, 24$ .

**Theorem 8.30** *If a  $BIBD(v, 6, \lambda)$  exists, then a  $BRD(v, 5, 4\lambda)$  exists.*

**Proof:** Take the BIBD as the master design in WFC, give each point a weight of 1, and use the  $BRD(6, 5, 4)$  of Example 8.1 as the ingredient design. In this case, this construction just amounts to replacing each block by the  $BRD(6, 5, 4)$  whose point set equals the points of the block.  $\square$

### Corollary 8.31

- 1) *Since  $BIBD(v, 5, \lambda)$  exists for every  $\lambda$  and  $v$  except  $BIBD(15, 5, 2)$ , then it follows that a  $BRD(v, 5, 8\lambda)$  and a  $BRD(v, 5, 12\lambda)$  exist for every  $\lambda$  and  $v$ ;*
- 2) *Since necessary conditions (i)  $\lambda(v - 1) \equiv 0 \pmod{5}$  and (ii)  $\lambda v(v - 1) \equiv 0 \pmod{30}$  are sufficient for the existence of  $BIBD(v, 6, \lambda)$  except 53 unknown  $v$  cases when  $\lambda = 1$  and  $BIBD(21, 6, 2)$ , hence they are sufficient for the existence of  $BRD(v, 5, 4\lambda)$ .*

**Proof :** In [66], Hanani showed the necessary conditions for BIBDs to be sufficient for the following values of  $k, v$  and  $\lambda$  :

- $k \leq 5$ , every  $\lambda, v$  except  $(v, k, \lambda) = (15, 5, 2)$  which is non-existent.
- $k = 6, \lambda \neq 1$ , every  $v$  except  $(v, k, \lambda) = (21, 6, 2)$  which is non-existent.

Since  $BIBD(v, 5, \lambda)$  exist for every  $\lambda$  and  $v$  except  $BIBD(15, 5, 2)$ , so  $BRD(v, 5, 8\lambda)$ 's and  $BRD(v, 5, 12\lambda)$ 's exist.

In ([3], 1.3), Abel showed that BIBDs exist for  $k = 6, \lambda = 1$  with 36 exceptions (3 impossible). In [83, I.1.28], parameter tables of known  $BIBD(v, k, \lambda)$ 's are given.

Since BIBDs for  $k = 6, \lambda > 1$  all exist except  $BIBD(21, 6, 2)$  and for  $\lambda = 1$  most exist, so it implies that  $BRD(v, 5, 4\lambda)$  exists for many  $v$  and  $\lambda$ .  $\square$

### Corollary 8.32

- 1) A  $BRD(v, 5, 4)$  exists for every  $v \equiv 1, 6 \pmod{15}$  and  $v > 801$ .
- 2) A  $BRD(v, 5, 4)$  exists for every  $v \equiv 1, 6 \pmod{15}$  and  $31 \leq v \leq 801$  except at most 34 cases.
- 3) A  $BRD(v, 5, 8)$  exists for every  $v \equiv 1, 6 \pmod{15}$  except  $v = 21$ .
- 4) A  $BRD(v, 5, 12)$  exists for every  $v \equiv 1 \pmod{5}$  and  $v \geq 11$ .
- 5) A  $BRD(v, 5, 20)$  exists for every  $v \equiv 0, 1 \pmod{3}$  and  $v \geq 7$ .
- 6) A  $BRD(v, 5, 60)$  exists for every  $v \geq 7$ .

**Remark 8.5.5** The above corollary follows from Theorem 8.29 and following results for the existence of  $BIBD(v, 6, \lambda)$ 's given in ([34], I.2.3).

- 1) A  $BIBD(v, 6, 1)$  exists for every  $v \equiv 1, 6 \pmod{15}$  and  $v > 801$ .
- 2) A  $BIBD(v, 6, 1)$  exists for every  $v \equiv 1, 6 \pmod{15}$  and  $31 \leq v \leq 801$  except 34 unknown cases.
- 3) A  $BIBD(v, 6, 2)$  exists for every  $v \equiv 1, 6 \pmod{15}$  except  $v = 21$ .
- 4) A  $BIBD(v, 6, 3)$  exists for every  $v \equiv 1 \pmod{5}$  and  $v \geq 11$ .

5) A  $BIBD(v, 6, 5)$  exists for every  $v \equiv 0, 1 \pmod{3}$  and  $v \geq 7$ .

6) A  $BIBD(v, 6, 15)$  exists for every  $v \geq 7$ .

**Corollary 8.33**

1) A  $BRD(v, 5, 8)$  and  $BRD(v, 5, 12)$  exists for every  $v \equiv 1, 5 \pmod{20}$  and  $v \geq 21$ .

2) A  $BRD(v, 5, 16)$  and  $BRD(v, 5, 24)$  exists for every  $v \equiv 1, 5 \pmod{10}$  and  $v \geq 11$  except  $v = 15$ .

3) A  $BRD(v, 5, 32)$  and  $BRD(v, 5, 48)$  exists for every  $v \equiv 0, 1 \pmod{5}$  and  $v \geq 6$ .

4) A  $BRD(v, 5, 40)$  and  $BRD(v, 5, 60)$  exists for every  $v \equiv 1 \pmod{4}$  and  $v \geq 9$ .

5) A  $BRD(v, 5, 80)$  and  $BRD(v, 5, 120)$  exists for every  $v \equiv 1 \pmod{2}$  and  $v \geq 7$ .

6) A  $BRD(v, 5, 160)$  and  $BRD(v, 5, 240)$  exists for every  $v \geq 6$ .

**Remark 8.5.6** The above corollary also follows from Theorem 8.29 and following results for the existence of  $BIBD(v, 5, \lambda)$ 's given in ([34], I.2.3).

1) A  $BIBD(v, 5, 1)$  exists for every  $v \equiv 1, 5 \pmod{20}$  and  $v \geq 21$ .

2) A  $BIBD(v, 5, 2)$  exists for every  $v \equiv 1, 5 \pmod{10}$  and  $v \geq 11$  except  $v = 15$ .

3) A  $BIBD(v, 5, 4)$  exists for every  $v \equiv 0, 1 \pmod{5}$  and  $v \geq 6$ .

4) A  $BIBD(v, 5, 5)$  exists for every  $v \equiv 1 \pmod{4}$  and  $v \geq 9$ .

5) A  $BIBD(v, 5, 10)$  exists for every  $v \equiv 1 \pmod{2}$  and  $v \geq 7$ .

6) A  $BIBD(v, 5, 20)$  exists for every  $v \geq 6$ .

**Lemma 8.34** : A  $BRD(v, 5, \lambda)$  exists for  $v = 6, 10, 11, 15, 20, 30, 31, 40, 51, 55, 66, 75, 76, 91, 96, 101, 106, 111, 121, 126, 136, 151, 156, 171, 181, 186$  and every  $\lambda \equiv 0 \pmod{4}$ .

**Remark 8.5.7**  $BRD(6, 5, 4)$ ,  $BRD(10, 5, 4)$ ,  $BRD(11, 5, 4)$ ,  $BRD(15, 5, 4)$ ,  $BRD(20, 5, 4)$ ,  $BRD(30, 5, 4)$  and  $BRD(40, 5, 4)$  are given in Examples 8.1, 8.10, 8.2, 8.13, 8.17, 8.19 and 8.21 respectively.

$BRD(v, 5, 4)$  exists for  $v = 31, 51, 55, 66, 75, 76, 91, 96, 101, 106, 111, 121, 126, 136, 151, 156, 171, 181, 186$  as per corollaries given above and BIBDs table in [83, I.1.28]. All their multiples also exist.

**Lemma 8.35** : *A  $BRD(v, 5, 4\lambda)$  exists for  $v = 16, 21, 25, 26, 36, 41, 45, 46, 61, 65, 81, 85, 105, 125, 141, 145, 161, 165$  and every  $\lambda \geq 2$ .*

**Remark 8.5.8**  $BRD(v, 5, 8)$  and  $BRD(v, 5, 12)$  exist for  $v = 16, 21, 25, 26, 36, 41, 45, 46, 61, 65, 81, 85, 105, 125, 141, 145, 161$  and  $165$  as per Corollary 8.31 and BIBDs table in [83, I.1.28]. All their multiples also exist. So  $BRD(v, 5, 4\lambda)$  exists for  $v = 15, 16, 21, 25, 26, 36, 41, 45, 46, 51, 61, 65, 81, 85, 101, 105, 125, 141, 145, 161, 165$  and every  $\lambda \geq 2$ .

**Lemma 8.36** : *A  $BRD(v, 5, 8\lambda)$  exists for  $v = 35, 71$  and every  $\lambda \geq 2$ .*

**Remark 8.5.9**  $BRD(v, 5, 16)$  and  $BRD(v, 5, 24)$  exist for  $v = 35, 71$  as per corollary 8.31 and BIBDs table in [83, I.1.28]. All their multiples also exist. Hence  $BRD(v, 5, 8\lambda)$  exists for  $v = 35, 71$  and every  $\lambda \geq 2$ .

We will now deal with  $BRD(v, 5, 20)$ . The necessary conditions do not restrict  $v$  except that  $v \geq 5$ . We will first need a corollary to Theorem 8.30.

**Corollary 8.37** *If  $v \equiv 0$  or  $1 \pmod{3}$ , and  $v \geq 6$ , then a  $BRD(v, 5, 20)$  exists.*

We next need a preliminary lemma on pairwise balanced designs taken from [14, III.3.2].

**Lemma 8.38** *If  $v > 34$  or  $v = 26$ , then a  $PBD(v, \{5, 6, 7, 8, 9\}, 1)$  exists.*

**Corollary 8.39** *If  $v > 34$  or  $v = 26$ , a  $BRD(v, 5, 20)$  exists.*

**Proof:** Apply Theorem 8.16 with the PBD as the master design, with each point of the BIBD receiving weight 1. Use  $BRD(v', 5, 20)$  for the ingredient designs, where these are given by Lemma 8.5.3 for  $v' = 5$ , by Corollary 8.37 for  $v' = 6, 7, 9$  and by Example 8.9 for  $v' = 8$ .  $\square$

**Theorem 8.40** *If  $v \geq 5$  and  $v \neq 32$ , a  $BRD(v, 5, 20)$  exists.*

**Proof:** By Corollary 8.37, we only have to deal with the 2 mod 3 values in the range 10 through 34 except 26. Constructions for  $v = 11, 14, 17, 20$  are given in Examples 8.2, 8.12, 8.16 and 8.17. For  $v = 23$ : use a  $BIBD(23, 11, 5)$  (from [83, I.1.3]) as the master design with weight 1 in WFC, with a  $BRD(11, 5, 4)$  as ingredient, to get a  $BRD(23, 5, 20)$ . For 29, we have a  $BRGDD_2(8, 4^8)$  by Theorem 8.23; removing a group gives a  $BRGDD_2(7, 4^7)$  of type  $4^7$ . Using this as the master in WFC with weight 1 and a  $BIBD(7, 5, 10)$  as ingredient gives a  $BRGDD_{20}(5, 4^7)$  of type  $4^7$ . Fill the groups with an extra point and  $BRD(5, 5, 20)$ 's.  $\square$

**Theorem 8.41** *A  $BRD(32, 5, 20t)$  exists for all  $t > 1$ .*

**Proof.** It suffices to establish this for  $\lambda = 40$  and  $60$ . For  $\lambda = 40$ , a  $BRD(31, 4, 2)$  exists [45, Theorem 4.1.1]; adjoin  $-\infty$  to one copy of this  $BRD$ , and  $+\infty$  to another copy; add 9 copies of the blocks of a  $BRD(31, 5, 4)$ . For  $\lambda = 60$ , use a  $BIBD(32, 6, 15)$  from Theorem 8.5.1, in Theorem 8.30.  $\square$

## 8.6 $BRD(10, 5, \lambda)$

There exists twenty one (21) inequivalent  $BIBDs$  with parameters  $(10, 5, 4)$ , these are given in [58], Appendix *B*. In this section, we construct thirteen (13)  $BRD(10, 5, 4)$ s from the  $BIBD(10, 5, 4)$ 's: call this set  $A$ , these are given in the Appendix *D* in the same order as in [58]. Gibbons  $BIBD$  numbers 9, 12, 13, 14, 15, 16, 19 and 21 were found, by an exhaustive computer search, not to give  $BRDs$ : call this set  $B$ . Hence thirteen of the twenty one inequivalent  $BIBD(10, 5, 4)$ s can be signed to  $BRD(10, 5, 4)$ s and eight cannot.

We write  $D_i || D_j = [D_i D_j]$  for the matrix of order  $v \times 2b$  with parameters  $(v, 2b, 2r, k, 2\lambda)$  when  $D_i$  is the matrix of order  $v$  with parameters  $(v, b, r, k, \lambda)$ . We note though that the eight  $BIBD(10, 5, 4)$ s, which can not be signed to  $BRD(10, 5, 4)$ s, satisfy Lemma 8.22. We use  $D_i || \overline{D_i}$  for the  $BRDs$  constructed using Lemma 8.22. Now we construct the  $BRD(10, 5, 8)$ s in two ways :

- i) if  $D_i, D_j \in A$ , there are 78 different  $BIBD(10, 5, 8)$ 's for  $D_i || D_j$ , so we obtain 78 inequivalent  $BRD(10, 5, 8)$ s;
- ii) if  $E_k \in B$ , then  $E_k || E_k$  is a  $BIBD(10, 5, 8)$ , so using Lemma 8.22, we obtain 8  $BRD(10, 5, 8)$ s.

Thus we have constructed 86  $BRD(10,5,8)$ 's of the  $\geq 135922$  possible cases.

**Theorem 8.42** *The conditions*

$$i) \lambda(v-1) \equiv 0 \pmod{4}$$

$$ii) \lambda v(v-1) \equiv 0 \pmod{20}$$

$$iii) b \equiv 0 \pmod{2}$$

$$iv) \lambda \equiv 0 \pmod{4}$$

*are necessary and sufficient for the existence of  $BRD(10, 5, \lambda)$ .*

**Remark 8.6.1** The inequivalent  $BRD(10,5,4)$ 's are given in the Appendix D.

## 8.7 $BRD(v, 5, 4)$

The other value that the existence of BIBDs with  $k = 6$  helps us with is  $\lambda = 4$ , but this only covers about one third of the values, and misses most of the smaller, more useful designs (it gives us  $v = 31, 66, 76, 91$ , etc.). The other small ( $\leq 45$ )  $BRD(v, 5, 4)$  are  $v = 6$  in Example 8.1,  $v = 11$  in Example 8.2,  $v = 25, 45$  in Theorem 8.43, and  $v = 10, 20, 21, 30, 31, 40, 41$  given in Section 8.9.

The necessary condition for the existence of a  $BRD(v, 5, 4)$  is that  $v \equiv 0$  or  $1 \pmod{5}$ . The objective of this section is to show this condition is sufficient, with the possible exception of 10 values of  $v$ , and the definite exception of  $v = 5$ . In order to do this, we exploit a number of previously constructed designs of various types that are available in the literature.

**Example 8.5** There exists a  $BRD(10,18,9,5,4)$ .

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & - & - & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & - & - & 0 & 1 & 0 & 0 & 0 & 0 & - & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & - & 0 & 1 & - & 0 & 0 & 0 & - & - & 0 & 0 & 1 & - & - & 0 \\ 1 & 0 & 0 & - & 0 & 0 & 0 & 1 & - & 0 & 0 & 0 & - & - & - & 1 & 0 & - \\ 0 & 1 & 0 & 0 & 1 & - & - & 0 & 0 & 0 & - & 0 & 1 & - & - & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & - & - & 0 & 1 & - & - & 0 & 0 & - & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & - & - & 0 & 1 & 0 & 1 & - & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & - & 0 & 1 & - & 1 & - & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & - & 0 & - & - & 0 & 1 & 0 & 1 & 0 & 0 & - & - \end{bmatrix}$$

**Example 8.6** There exists a  $BRD(11,22,10,5,4)$ .



$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & - & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & - & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & - & 0 & 0 & - & 0 & 0 & 1 & 1 & 0 & 1 & 0 & - & 0 & 0 & - & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & - & 0 & 0 & - & 0 & - & 0 & 1 & 1 & 0 & 0 & - & 0 & 0 & - & 0 & 1 \\ 1 & 0 & 0 & 0 & - & 0 & 0 & - & 0 & - & 1 & 0 & 0 & 0 & - & 0 & 0 & - & 0 & - \\ 0 & 1 & - & 0 & 0 & 0 & 0 & 1 & - & 0 & - & 0 & 1 & - & 0 & 0 & 0 & 1 & - & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & - & 0 & 0 & - & 0 & 0 & - & 0 & - & 1 & 0 \\ 0 & 1 & 0 & - & 0 & 1 & 0 & 0 & 0 & 1 & - & 0 & - & 0 & 1 & 0 & - & 0 & 0 & - \\ 0 & 0 & - & 0 & 1 & 1 & - & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & - & - & 1 & 0 & 0 \\ 0 & 0 & 1 & - & 0 & 0 & - & 1 & 0 & - & 0 & 0 & 0 & - & 1 & 0 & 0 & 1 & - & 0 \\ 0 & 0 & 0 & 1 & - & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & - & 1 & - & 0 & - & 0 & 0 \end{bmatrix}$$

**Theorem 8.43** Define  $E = \{5, 11, 15, 35, 71, 75, 85, 95, 111, 115, 135, 195, 215, 335\}$ . If  $n \equiv 1, 5 \pmod{10}$  and  $n \notin E$ , then there is a  $GDD(5, 2^n)$ .

**Proof:** See [114, Theorem 1.2]. □

**Corollary 8.44** Define  $E = \{5, 11, 15, 35, 71, 75, 85, 95, 111, 115, 135, 195, 215, 335\}$ . If  $v \equiv 1, 5 \pmod{10}$  and  $v \notin E$ , then there is a  $BRD(v, 5, 4)$ .

**Proof.** Essentially this is an application of Theorem 8.25 to Theorem 8.43. □

**Theorem 8.45** If  $q \geq 5$ , then there is a  $GDD(5, 20^q)$ .

**Proof:** See [114, Theorem 1.2]. □

**Corollary 8.46** If  $q \geq 5$ , then there is a  $BRGDD(5, 10^q)$ , and a  $BRD(v, 5, 4)$  exists for any  $v \equiv 0, 1 \pmod{10}$  if  $v \geq 50$ .

**Proof:** The  $BRGDD$  follows by Theorem 8.25 and the  $BRD$  follows from Theorem 8.18.

**Remark 8.7.1** From [6, II.2.73], if  $m \geq 5$ , and  $m \notin \{6, 10, 14, 18, 22\}$ , then a  $TD(6, m)$  exists, and if  $m \neq 10$ , then a  $TD(5, m)$  exists.

**Theorem 8.47** If a  $TD(5, 2m)$  exists, then a  $BRGDD_4(5, m^5)$  exists. If furthermore, a  $BRD(m + w, 5, 4)$  exist for some  $w \in \{0, 1\}$ , then a  $BRD(10m/2 + w, 5, 4)$  exists.

**Proof:** We collapse points using Theorem 8.25 to get the  $BRGDD$ , then use Theorem 8.18 to fill the groups in with the aid of  $w$  new points. □

**Corollary 8.48** A  $BRGDD(46, 5, 4)$  and a  $BRGDD(146, 5, 4)$  exist.

**Proof.** Let  $m = 9$ ,  $w = 1$ , or  $m = 29$ ,  $w = 1$  in Theorem 8.47. □

**Theorem 8.49** *If a  $TD(6, m)$  exists and  $0 \leq n \leq m$ , then a  $GDD(\{5, 6\})$  of type  $m^5 n^1$  exists. If furthermore, a  $BRD(2m + w, 5, 4)$  and a  $BRD(2n + w, 5, 4)$  both exist for some  $w \in \{0, 1\}$ , then a  $BRD(10m + 2n + w, 5, 4)$  exists.*

**Proof:** Truncate one group of the TD to size  $n$  to get the GDD. Use this as the master design in Theorem 8.16 with all points getting a weight of 2. Finally, add  $w$  new points, and fill in the groups with the BRDs.  $\square$

**Corollary 8.50** *A  $BRGDD(236, 5, 4)$  exists.*

**Proof.** Let  $m = 23$ ,  $n = 3$ ,  $w = 0$  in Theorem 8.49.  $\square$

**Lemma 8.7.2** *A  $BRGDD(36, 5, 4)$  exists.*

**Proof.** Use a  $BRGDD_2(5, 2^6)$  as the master design. In Theorem 8.16, give each point a weight of 3, and use a  $TD_2(5, 3)$  from [66, Theorem 3.11] as the ingredient, to get a  $BRGDD_4(5, 6^6)$ , which can be filled with a  $BRD(6, 5, 4)$  by Theorem 8.17.  $\square$

**Theorem 8.51** *Let  $v = 20t + 5$ . If  $v \notin \{45, 225, 345, 465, 645\}$  (i.e.,  $t \notin E = \{2, 11, 17, 23, 32\}$ ), then a  $RBIBD(v, 5, 1)$  exists.*

**Proof:** See [9], updated in [7].  $\square$

**Corollary 8.52** *Let  $n \leq 5t$ , and  $t \notin \{2, 11, 17, 23, 32\}$ , and  $w \in \{0, 1\}$ . If a  $BRD(2n + w, 5, 4)$  exists, then a  $BRD(40t + 10 + 2n + w, 5, 4)$  exists.*

**Proof:** We take the  $RBIBD(20t + 5, 5, 1)$ , and remove a parallel class to provide  $4t + 1$  groups of size 5, then add new points to  $n$  of the remaining parallel classes, to give a  $GDD_1(\{5, 6\}, 5^{4t+1} n^1)$ . We use this as the master design in WFC, and give every point a weight of 2, thus a  $BRGDD_4(5, 10^{4t+1} 2n^1)$  exists, using ingredient designs from Corollary 8.24 or Example 8.23. We then fill the groups, using  $w$  new points and Theorem 8.18. Note that a  $BRD(10 + w, 5, 4)$  exists by Example 8.10 or 8.2. and for the  $i$ -th group,  $G_i$ , we add the blocks of a  $BRD(|G_i| + w, 5, 4)$  on the points  $G_i$  and  $w$ ; doing this for every group yields the required BRD.  $\square$

**Corollary 8.53** *A  $BRD(v, 5, 4)$  exists for  $v \in \{195, 335\}$ .*

**Proof:** Take  $t = 4$ ,  $n = 12$  and  $w = 1$ , or  $t = 7$ ,  $n = 22$  and  $w = 1$  in Corollary 8.52.

We now use this corollary to construct some  $BRD(v, 5, 4)$ s for the  $v \equiv 6 \pmod{10}$  cases, noting that we already have  $BRD(v, 5, 4)$ 's for  $v = 6, 10, 11, 20, 30, 31, 36, 40, 46, 66, 76$ .

mod	$n$	$w$	first	$2n + w$	first	Later exceptions
40			$t$		$v$	
6	18	0	4	36	206	(486),(726),(966),(1326)
16	3	0	1	6	56	(96),(456),(696),(936),(1296)
26	28	0	6	56	306	506,746,986,1346
	48	0	10	96	506	(546),(786),(1026),(1386)
36	33	0	7	66	356	(516),(756),(996),(1356)

The parenthesised exceptions are all  $v$  for which a  $BIBD(v, 6, 1)$  exists. We know a  $BIBD(v, 6, 1)$  also exists for  $v = 66, 76, 106, 126, 156, 186, 196, 276$ . For these values a  $BRD(v, 5, 4)$  exists by Theorem 8.30. We have given two constructions for the 26 mod 40 case, so the exceptions from one can be covered by the other.

**Lemma 8.54** *If  $v \in \{115, 116, 166, 206, 226, 266, 316\}$ , then there exists a  $PBD(v, \{6, 10, 11, 20, 21, 31, 41\}, 1)$ , and consequently a  $BRD(v, 5, 4)$  exists.*

**Proof:** For  $v = 116$ : delete 5 collinear points of  $AG(2, 11)$ . For  $115 = 6 * 19 + 1$ : fill a  $TD(6, 19)$  with an extra point. For  $v = 166, 226, 266$ : see [9, Lemma 3.2] For  $v = 206$ : complete the  $RBIBD(165, 5, 1)$  to give a  $PBD(\{6, 41\})$ . For  $316 = 10 * 31 + 6$ : truncate a group of a  $TD(11, 31)$ . □

The following theorem is the main result of this section.

**Theorem 8.7.3** *If  $v \equiv 0, 1 \pmod{5}$ , then a  $BRD(v, 5, 4)$  exists with the definite exception of  $v = 5$ , and the possible exception of 10 further values of  $v$ , the largest of which is 215. These values are given in Table 8.7.1.*

**Table 8.7.1**  
Table of  $v$  with  $BRD(v, 5, 4)$  unconstructed.

(5)	16	26	35	75	85	86	95	135	215
-----	----	----	----	----	----	----	----	-----	-----

We next address the existence of  $BRD(v, 5, \lambda)$  for these exceptional values of  $v$  with  $\lambda > 4$ .

**Lemma 8.55** *If  $v \equiv 0, 1 \pmod{5}$  and  $v \notin \{10, 11, 15, 16, 20, 35, 40, 50, 51, 80\}$ , then a  $PBD(v, \{5, 6\}, 1)$  exists.*

**Proof:** See [14, III.3.17]. □

**Lemma 8.56** *If  $v \equiv 0, 1 \pmod{5}$  and  $v \notin \{15, 16, 35\}$ , then a  $BRD(v, 5, 4t)$  exists for all  $t > 1$ .*

**Proof:** Since we know a  $BRD(5, 5, 4t)$  exists for all  $t > 1$ , and a  $BRD(6, 5, 4)$  exists, the result follows from Lemma 8.55, after removing from the exception set there those  $v$  for which we have constructed a  $BRD(v, 5, 4)$ . □

**Theorem 8.7.4** *If  $v \equiv 0, 1 \pmod{5}$ , then a  $BRD(v, 5, 4t)$  exists for all  $t > 1$ , with the possible exception of  $BRD(15, 5, 12)$ ,  $BRD(35, 5, 8)$ ,  $BRD(35, 5, 12)$  and  $BRD(35, 5, 28)$ .*

**Proof:** We have a  $BRD(v, 5, 20)$  by Theorem 8.40. We have a  $BRD(15, 5, 8)$  by Example 8.14. Using the  $BIBD(16, 6, t)$  for all  $t > 1$ , given by Theorem 8.5.1, we have, by Theorem 8.30, that a  $BRD(16, 5, 4t)$  exists for all  $t > 1$ . Using the  $BIBD(35, 5, 2)$  given by Theorem 8.5.1, and so a  $BRD(35, 5, 16)$ , and a  $BRD(35, 5, 24)$  by Theorem 8.29. □

## 8.8 $BRD(v, 5, 10)$

In this section, we examine the case of  $\lambda = 10$ . The key basic construction is Theorem 8.9.1, where  $BRD(v, 5, 10)$  are constructed for all prime powers,  $v > 5$ , with  $v \equiv 1 \pmod{4}$ .

**Lemma 8.57** : *A  $BRD(v, 5, \lambda)$  exists for  $v = 13, 17$  and every  $\lambda \equiv 0 \pmod{10}$ .*

**Example 8.7** There exists a  $BRD(13, 78, 30, 5, 10)$  for a  $BIBD(13, 78, 30, 5, 10)$  constructed from the  $2 \times BIBD(13, 39, 15, 5, 5)$ . The  $BIBD(13, 39, 15, 5, 5)$  is constructed cyclically from the following base blocks :

$$[0, 1, 2, 4, 8]; [0, 1, 3, 6, 12]; [0, 2, 5, 6, 10] \pmod{13}.$$

Its  $BRD(13, 78, 30, 5, 10) = X + Y$  is given below :

$$X = \begin{bmatrix} 100001000101111000001001011001000110010 \\ 110000100010111100000100100100100011001 \\ 11100001000100111000001001\bar{1}01001000\bar{1}100 \\ 011100001000110111000001000\bar{1}0\bar{1}001000\bar{1}\bar{1}0 \\ 10111000010000101\bar{1}1000001000\bar{1}0100\bar{1}0001\bar{1} \\ 0101110000100001011100000\bar{1}10010\bar{1}00\bar{1}0001 \\ 0010111000010100\bar{1}011\bar{1}00000\bar{1}\bar{1}00\bar{1}0\bar{1}001000 \\ 00010111000010\bar{1}0010\bar{1}1\bar{1}00000110010100100 \\ 100010111000000100\bar{1}0\bar{1}\bar{1}100000\bar{1}\bar{1}0010\bar{1}0010 \\ 010001011\bar{1}000000\bar{1}00\bar{1}011\bar{1}000001\bar{1}00\bar{1}0\bar{1}00\bar{1} \\ 0010001011\bar{1}000000\bar{1}00101\bar{1}\bar{1}01000\bar{1}\bar{1}0010\bar{1}00 \\ 00010001011\bar{1}000000\bar{1}00\bar{1}0\bar{1}110\bar{1}000\bar{1}\bar{1}0010\bar{1}0 \\ 000010001011\bar{1}\bar{1}0000010010\bar{1}10010001\bar{1}0010\bar{1} \end{bmatrix}$$

$$Y = \begin{bmatrix} 100001000101111000001001011001000110010 \\ \bar{1}10000100010\bar{1}\bar{1}\bar{1}1000001001001001000\bar{1}1001 \\ 1\bar{1}100001000\bar{1}00\bar{1}\bar{1}100000100\bar{1}\bar{1}01001000\bar{1}100 \\ 01\bar{1}100001000110\bar{1}\bar{1}100000\bar{1}000\bar{1}0\bar{1}001000\bar{1}\bar{1}0 \\ \bar{1}01\bar{1}10000\bar{1}000010\bar{1}\bar{1}100000\bar{1}000\bar{1}0\bar{1}00\bar{1}000\bar{1}\bar{1} \\ 0\bar{1}01\bar{1}\bar{1}0000\bar{1}000010\bar{1}\bar{1}100000\bar{1}\bar{1}0010\bar{1}001000\bar{1} \\ 00\bar{1}011\bar{1}0000\bar{1}0\bar{1}0010\bar{1}\bar{1}\bar{1}000001\bar{1}00\bar{1}0\bar{1}001000 \\ 000\bar{1}0\bar{1}\bar{1}\bar{1}0000\bar{1}010010\bar{1}\bar{1}\bar{1}00000\bar{1}\bar{1}00\bar{1}0\bar{1}00100 \\ 1000\bar{1}0\bar{1}\bar{1}\bar{1}000000100101\bar{1}\bar{1}000001\bar{1}00\bar{1}0\bar{1}0010 \\ 0100010\bar{1}\bar{1}1000000\bar{1}00101\bar{1}\bar{1}000001\bar{1}00\bar{1}0\bar{1}001 \\ 001000\bar{1}0\bar{1}\bar{1}\bar{1}000000100\bar{1}01110\bar{1}0001\bar{1}00\bar{1}0\bar{1}00 \\ 000\bar{1}000101\bar{1}1000000\bar{1}00\bar{1}0\bar{1}11010001\bar{1}00\bar{1}0\bar{1}0 \\ 0000\bar{1}0001011\bar{1}\bar{1}00000\bar{1}00\bar{1}01100\bar{1}0001100\bar{1}0\bar{1} \end{bmatrix}$$

**Example 8.8** There exists a BRD(17,136,40,5,10) constructed from two copies of a BIBD(17, 68, 20, 5, 5). The base blocks of the BIBD are:

$$[0, 1, 4, 13, 16]; [0, 3, 5, 12, 14]; , [0, 2, 8, 9, 15]; [0, 6, 7, 10, 11] \pmod{17}.$$

Its BRD(17,136,40,5,10) = X + Y + Z is given below :

$$X = \begin{bmatrix} 110010000000010011001010000001010010100000110 \\ 1110010000000001000100101000000101001010000011 \\ 0111001000000000100010010100000010110101000001 \\ 0011100100000000011001001010000001001010100000 \\ 100111001000000000100100101000000100101010000 \\ 01001110010000000010100\bar{1}00101000000000\bar{1}0101000 \\ 001001110010000000010100\bar{1}00101000000000\bar{1}0\bar{1}0100 \\ 0001001110010000000010100\bar{1}00101000000000\bar{1}0\bar{1}010 \\ 00001001110010000000010100\bar{1}00\bar{1}01000\bar{1}0000010\bar{1}01 \\ 000001001110010000000010100\bar{1}0010\bar{1}001\bar{1}0000010\bar{1}0 \\ 000000100111001000000001010010010\bar{1}001\bar{1}00000\bar{1}0\bar{1} \\ 000000010011100100000001010010010100\bar{1}\bar{1}0000010 \\ 000000001001110011000000010100\bar{1}0010000\bar{1}\bar{1}00000\bar{1} \\ 1000000001001110001000000010\bar{1}00\bar{1}0010000\bar{1}\bar{1}00000 \\ 0100000000100111010\bar{1}000000\bar{1}0\bar{1}00100000000\bar{1}\bar{1}0000 \\ 0010000000001001110\bar{1}01000000\bar{1}0\bar{1}00\bar{1}0\bar{1}00000\bar{1}1000 \\ 10010000000001001\bar{1}0010\bar{1}00000010\bar{1}00\bar{1}0\bar{1}00000\bar{1}\bar{1}00 \end{bmatrix}$$



**Lemma 8.60** *A  $GDD(5, n^{4t+1})$  exists for  $n \in \{5, 15\}$  for all  $t$ .*

**Proof:** See [114]. □

**Lemma 8.61** *A  $BRD(v, 5, 2)$  exists whenever:*

1.  $v \equiv 41 \pmod{160}$ .
2.  $v \equiv 61 \pmod{240}$ .
3.  $v \equiv 81 \pmod{320}$ .

**Proof:** Use the GDD given in Lemma 8.60 as the master design in Theorem 8.16, and give each point in 5-groups weight of 8 or 16, and give each point in 15-groups weight of 4. Use Corollary 8.24 to give the needed ingredient design, and fill this design with BRDs using an additional point and Theorem 8.18, with the filling designs from Theorem 8.26. □

**Lemma 8.8.1** *A  $BRD(v, 5, 10)$  exists for  $v \in \{161, 321, 545\}$ .*

**Proof:** For  $v = 161, 321$ ; remove all the blocks through a point, and use these to define groups of a  $GDD(5, 4^n)$  for  $n = 10, 20$ . use this as the master design in WFC, and give each point a weight of 4, then fill the groups using a  $BRD(17, 5, 10)$ , with an extra point and Theorem 8.18. Similar to Corollary 8.52, remove a parallel class, and add 3 points to a  $RBIBD(65, 5, 1)$  to get a  $GDD(\{5, 6\}, 5^{13}3^1)$ , then give each point a weight of 8 in WFC, use the ingredients of Lemma 8.59 to give a  $BRGDD_2(5, 40^{13}24^1)$ , and then fill the groups. □

**Lemma 8.62** *The following  $BRD(v, 5, 10)$  designs with a  $BRD(u, 5, 10)$  subdesign exist:*

- i).  $u \in \{9, 17\}$  and  $v = 145$ .
- ii).  $u = 9$  and  $v \in \{41, 49\}$ ;
- iii).  $u = 13$  and  $v \in \{61, 69, 73\}$ ;
- iv).  $u = 17$  and  $v \in \{81, 97\}$ .

**Proof:** the case  $v = 145$  follows from Lemma 8.58. For  $v = 5(u - 1) + 1$ , use a  $BRGDD_2(5, 4^5)$  as the master design, and give each point a weight of  $w = (v - 1)/20$ , and use a  $TD_5(5, w)$  as the ingredient design, then fill with the aid of one additional point. The subdesign is the last filling design. For  $v = 6(u - 1) + 1$ , use a  $BRGDD_2(5, 2^6)$  as the master design, and give each point a weight of  $w = (v - 1)/12$ , and use a  $TD_5(5, w)$  as the ingredient design, then fill with the aid of one additional point. The subdesign is the last filling design. For the  $BRD(69, 5, 10)$ ; this is constructed using Theorem 8.8.2 below, by filling the groups of a  $BRGDD_{10}(5, 12^5 8^1)$  with an additional point. we can take one of these filling designs as the subdesign.  $\square$

**Theorem 8.8.2** *Suppose a  $BRD(4m + 1, 5, 1)$  and a  $BRD(4n + 1, 5, 1)$  both exist, and  $t = 5m + n$ , with  $0 \leq n \leq m$ ; then there exists a  $BRD(4t + 1, 5, 10)$ .*

**Proof:** Truncate one group of a  $TD_5(6, m)$  to size  $n$ , and use this as the master design in Theorem 8.16, giving points weight 4. This TD exists for all  $m$  by [66, Theorem 3.11]. Use the BRGDDs of Lemma 8.59 as the ingredient designs to get a  $BRGDD_{10}(5, (4m)^5(4n)^1)$ , and fill this design with BRDs using an additional point and Theorem 8.18.  $\square$

**Corollary 8.63** *A  $BRD(4t + 1, 5, 10)$  exists for  $t \in \{44, 46, 50, 53, 76, 86\}$ .*

**Proof.** For this variant of Theorem 8.8.2, we fill the designs with the aid of  $u$  additional points, where we have a  $BRD(4m + u, 5, 10)$  containing a  $BRD(u, 5, 10)$  subdesign, and a  $BRD(4n + u, 5, 10)$  design exists. We take  $m = 8$  with  $u = 9$ ,  $m = 12$  with  $u = 13$ ,  $m = 14$  with  $u = 13$ , or  $m = 16$  with  $u = 17$  to get the above constructions, where the BRDs with subdesigns follow from Lemma 8.62.  $\square$

**Theorem 8.8.3** *If  $v = 4t + 1 > 445$ , then a  $BRD(v, 5, 10)$  exists. For smaller  $v$ , there are at most 12 possible exceptions, as given in Table 8.8.1, with  $v = 5$  being a definite exception.*

Table 8.8.1

$t$	(1)	5	8	11	14	16	19	21	26	61
$4t + 1$	(5)	21	33	45	57	65	77	85	105	245
$t$	101	111								
$4t + 1$	405	445								



**Proof.** The existence of  $\text{BRD}(v, 5, 10)$  is established by application of Theorem 8.8.2, using the construction of Theorem 8.9.1 to deal with prime power cases, and with the help of Lemmas 8.58, 8.8.1 and Corollary 8.63. The non-existence result for  $v = 5$  was given in Lemma 8.5.3. For the larger values of  $t > 111$ , take  $n \in \{15, 36, 7, 18, 9\}$  in Theorem 8.8.2, and let  $v = 4t + 1$  with  $t = 5m + n$ ; if  $m$  is not valid, then try using  $m + 1$  and  $n - 5$ . This deals with all values of  $t \geq 108$ , except for  $t \equiv 1 \pmod 5$ . For these values we may use  $n \in \{6, 31\}$ , and deal with all values of  $5m + n > 111$ , except 136, which is covered by Corollary 8.63. The smaller values of  $t$ , (i.e.,  $t \leq 111$ ), can be easily checked.  $\square$

## 8.9 Construction of BRDs from SDSs

In this section, Examples of Bhaskar Rao designs constructed from SDS's will be given, some of which have been used in proofs of Sections 4 and 5. We first look at some simple non-existence results for signed SDS via counting arguments, and start by examining possible parameter sets of interest.

Table 8.9.1

$v$	$b$	$r$	$k$	$\lambda$	$ G $	$\rho$	Number of finite pairs in SDS
$20t + 1$	$2tv$	$10t$	5	2	$v$	0	$20t$
$20t + 5$	$(4t + 1)(10t + 2)$	$10t + 2$	5	2	?	?	?
$10t$	$2tv$	$10t - 1$	5	4	$v$	0	$20t$
$10t + 1$	$2tv$	$10t$	5	4	$v$	0	$20t$
$10t + 5$	$(2t + 1)(v - 1)$	$10t + 4$	5	4	$v - 1$	1	$20t + 6$
$10t + 6$	$(2t + 1)v$	$10t + 5$	5	4	$v$	0	$20t + 10$
$4t + 1$	$2tv$	$10t$	5	10	$v$	0	$20t$
$t + 1$	$tv$	$5t$	5	20	$v$	0	$10t$
$t + 1$	$(t + 1)(v - 1)$	$5t$	5	20	$v - 1$	5	$10t - 10$

Now we consider the  $\rho$  blocks containing a fixed element  $\infty$  (which we will consider to be of constant sign); suppose there are  $x_i$  blocks containing  $i$  positive elements with fixed element  $\infty$ . Suppose these  $\rho$  blocks contribute  $M$  mixed sign pairs in total. Then:

$$\rho(k - 1)/2 = \sum ix_i$$

$$M = \sum i(k-1-i)x_i$$

Now adding these gives  $M + 2\rho = \sum i(k-i)x_i$  for  $k = 5$ , and since  $k$  is odd,  $i(k-i)$  is always even, so  $M$  is even, whatever (valid) pattern of signing we pick. Also, for blocks that do not have the fixed point, we have  $i(k-i)$  which is always even, so we must always have an even number of mixed sign pairs, and thus the total number of finite pairs in the SDS must be a multiple of 4. This eliminates  $(v, 5, 4)$  SDS for  $v \equiv 1$  or  $5 \pmod{10}$ , (at least using full orbits over  $Z_v$  or  $Z_{v-1}$ ) for these cases (and all their odd multiples), and determines that for  $(v, 5, 20)$  we should look at  $Z_{v-1}$  if  $v$  is even, and  $Z_v$  if  $v$  is odd. Also,  $(20t+5, 5, 2)$  is not an option with these groups.

We next consider a signed SDS for  $\text{BRD}(4t+1, 5, 10)$  in the case that  $4t+1 = q$  is a prime power. Let  $x$  be a primitive element for  $GF(q)$ .

$$\mathcal{P} = GF(q)$$

$$B_\alpha = \{0, -x^\alpha, x^{\alpha+2t}, bx^\alpha, -bx^{\alpha+2t}\} \pmod{q}$$

$$C_\alpha = \{0, x^\alpha + 1, x^{\alpha+2t+1}, bx^{\alpha+1}, -bx^{\alpha+2t+1}\} \pmod{q}$$

$$\text{for } \alpha = 0, 2, 4, \dots, 2t-2$$

The signed differences arising from  $B_0$  and  $C_0$  are:

$$\begin{array}{cccccccccccc} \pm 1 & +- & \pm b & +- & \pm(b+1) & ++ & \pm(b-1) & -- & \pm 2 & - & \pm 2b & - \\ \pm x & ++ & \pm bx & +- & \pm(b+1)x & +- & \pm(b-1)x & +- & \pm 2x & + & \pm 2bx & - \end{array}$$

The unbalanced elements are  $\pm x$  and  $\pm(b+1)$ , both with  $(++)$ , and  $\pm(b-1)$  and  $\pm 2$ , both with  $(--)$ . (If  $b$  is square,  $\pm 2x$  will be balanced by  $\pm 2bx$ , and by  $\pm 2b$  otherwise). We can usually achieve balance by a careful choice of  $b$ ; solutions were found for all prime powers through 125, (except 5 of course). It is easy to see that a solution must exist, at least for  $q = 4t+1 > 5$ : if  $\pm 2$  is a square, we require that  $b+1$  be a square, and  $b-1$  not be a square, so  $b^2-1$  is not a square; there are  $t$  values of  $b$  for which  $b^2-1$  is not a square (see [65, p. 178]), so either we have what we want, or else we have a value  $b'$  such that  $b'+1$  is not a square, and  $b'-1$  is; in this case  $b = -b'$  is our solution. Alternatively, if  $\pm 2$  is not a square, then it will cancel out the  $\pm x$  signs, and we need  $b-1$  and  $b+1$  to be of the same quadratic character, (i.e.,  $b^2-1$  is a square), so that they will cancel each other out; there are  $t-1$  such values of  $b^2$ , and so  $2t-2$  possible values of  $b$ ; (and no solutions for  $v = 5$ ). Note that  $b = \pm 1$  is not counted in any of this, since  $b^2-1 = 0$  then, so we do not have to worry about the requirement of having distinct elements in the block.

**Theorem 8.9.1** *If  $v = 4t + 1 > 5$  is a prime power, then a  $BRD(v, 5, 10)$  exists.*

We now give explicit solutions for the smaller prime powers:

**Table 8.9.2**

$q$	$x$	$b$	$\log(b-1)$	$\log(b)$	$\log(b+1)$	$\log(2)$
	(++)		(--)		(++)	(--)
9	$x^2 = 2x + 1$	$x + 1$	1	7	6	4
13	2	2	0	1	8	1
17	3	8	11	10	2	14
25	$x^2 = 4x + 3$	$x + 1$	1	17	14	6
29	2	5	2	22	6	1
37	2	2	0	1	26	1
41	6	4	15	12	22	26
49	$x^2 = 6x + 4$	$x + 3$	11	26	12	16
53	2	4	17	2	47	1
61	2	2	0	1	6	1
73	5	8	33	24	12	8
81	$x^4 = 2x^3 + 1$	$x + 1$	1	77	68	40
89	3	4	1	32	70	16
97	5	8	31	6	44	34
101	2	5	2	24	70	1
109	6	4	52	6	76	57
113	3	6	83	8	36	12
121	$x^2 = 10x + 4$	$x + 1$	1	71	68	36
125	$x^3 = 4x^2 + 3$	$x + 1$	1	29	99	93

We now give some explicit SDS's for small  $v$ .

**Example 8.9** There exists a  $BRD(8, 5, 20)$  consisting of the following base blocks:

$$[0, 1, 2, -3, 4]; [0, 1, 2, 4, -5]; [0, 1, 2, 4, -6]; [\infty, 0, -1, -2, -4];$$
$$[\infty, -0, 1, 2, 4]; [\infty, -0, -1, 2, 4]; [\infty, -0, 1, -2, 4]; [\infty, -0, 1, 2, -4] \pmod{7}$$

**Example 8.10** There exists a  $BRD(10, 18, 9, 5, 4)$ .

$$[-00, 01, 02, 11, 22]; [\infty, 01, 02, -11, -22] \pmod{(3, 3)}$$

**Example 8.11** There exists a  $\text{BRD}(13, 5, 10)$  constructed from the double of a  $\text{BIBD}(13, 5, 5)$  taken from [4, IV.10.9]. The base blocks are:

$$[0, 1, 2, -4, -8]; [0, -1, 2, -4, -8]; [0, 1, -3, 6, -12]; \\ [0, -1, 3, 6, 12]; [0, 2, 5, 6, -10]; [0, 2, 5, -6, 10] \pmod{13}$$

**Example 8.12** There exists a  $\text{BRD}(14, 5, 20)$  consisting of the blocks of the  $\text{BRD}(13, 5, 10)$  constructed in Example 8.11 above, augmented with the following base blocks:

$$[\infty, -1, -2, 4, 8]; [\infty, -3, -6, 11, 12]; [\infty, -5, 7, 9, -10]; [\infty, -0, 1, -3, 9]; \\ [-0, 1, 2, 4, 8]; [0, 3, 6, 11, 12]; [0, -5, 7, 9, -10]; [\infty, -0, 1, 3, -9] \pmod{13}$$

**Example 8.13** A  $\text{BRD}(15, 5, 4)$  was found from 3-(15, 5, 4)SDS taken from Hall [65] page 295. The base blocks of 3-(15, 5, 4)SDS are :

$$[0, 1, 4, 9, 11]; [0, 1, 4, 10, 12]; [\infty, 0, 1, 2, 7] \pmod{14}.$$

**Example 8.14** A  $\text{BRD}(15, 5, 8)$  was found from 6-(15, 5, 4) SDS taken from Hall [65, p. 410]. The base blocks of the  $\text{BRD}(15, 5, 8)$  are:

$$[-\infty, 0, -1, 2, -7]; [-\infty, 0, 1, -2, -7]; [0, -1, -4, 9, -11]; \\ [0, 1, -4, 9, -11]; [0, 1, 4, -10, -12]; [-0, -1, -4, -10, -12] \pmod{14}$$

**Example 8.15** A  $\text{BRD}(16, 5, 8)$  exists by Theorem 8.30. An alternative construction is given by the following 6-(16, 5, 8) SDS. The base blocks of the  $\text{BRD}(16, 5, 8)$  are:

$$[0, -1, 2, 4, -7]; [0, -1, -2, 4, 7]; [0, -1, 5, 8, -10]; \\ [0, 1, 5, -8, 10]; [0, 1, 3, 7, -11]; [0, 1, 3, -7, 11] \pmod{16}.$$

**Example 8.16** There exists a  $\text{BRD}(17, 5, 10)$  constructed from the double of a 4-(17, 5, 5) SDS. The base blocks of  $\text{BRD}(17, 5, 10)$  are:

$$[0, 1, 4, -13, -16]; [0, 1, -4, 13, -16]; [0, 3, 5, -12, -14]; [0, 3, -5, 12, -14]; \\ [0, 2, 8, 9, -15]; [0, 2, 8, -9, 15]; [0, 6, 7, 10, -11]; [0, 6, 7, -10, 11] \pmod{17}.$$

**Example 8.17**  $\text{BRD}(20, 5, 4)$  was constructed from the following base blocks:

$$[-1, -5, 6, -10, -12]; [4, -7, -8, -13, -16];$$

$$[9, 11, 15, -17, -18]; [\infty, 0, -2, 3, -14] \pmod{19}.$$

**Example 8.18** A  $\text{BRD}(21, 5, 4)$  adapted from Assaf's GDD in [114, Lemma 2.2]:

$$[0, 2, 5, 11, -4]; [0, -1, -3, -7, -12];$$

$$[0, 1, 8, -16, -19]; [0, 4, -9, -10, -17] \pmod{21}$$

**Example 8.19**  $\text{BRD}(30, 5, 4)$  was found from a  $6 - (30, 5, 4)$  SDS, the base blocks of  $\text{BRD}(30, 5, 4)$  are:

$$[-0, 1, 8, 10, 13]; [-0, -4, -15, -21, 26]; [-0, -1, 10, 14, \infty];$$

$$[-0, 3, 4, -11, 23]; [0, -2, 16, -17, 26]; [-0, 4, 6, 11, 27] \pmod{29}$$

**Example 8.20** A  $\text{BRD}(31, 5, 4)$  exists by Theorem 8.30. Using a difference set construction for a  $\text{BIBD}(31, 6, 1)$ , i.e.,  $[1, 5, 11, 24, 25, 27]$  yields the following  $6 - (31, 5, 4)$  SDS:

$$[1, 5, -11, -24, -25]; [1, -5, 11, -24, -27]; [-1, 5, 11, -25, -27];$$

$$[1, 5, 24, 25, -27]; [1, 11, 24, -25, 27]; [5, 11, -24, 25, 27] \pmod{31}.$$

**Example 8.21**  $\text{BRD}(40, 5, 4)$  was found from a  $8 - (40, 5, 4)$  SDS containing the base blocks:

$$[\infty, 0, 3, -9, -27]; [2, -5, -13, -26, -32]; [-1, 4, -20, -29, -36];$$

$$[-8, 16, 25, -30, -35]; [10, -11, -12, -14, -22]; [-7, 15, -17, -31, -33];$$

$$[6, 21, 28, -34, -38]; [18, 19, -23, 24, -37] \pmod{39}.$$

**Example 8.22** A  $\text{BRD}(41, 5, 2)$  adapted from [114, Lemma 2.1].

$$[-0, 1, 3, 7, -34]; [-0, 5, -16, -30, -29];$$

$$[-0, -8, 23, 2, 20]; [-0, 1, 19, 4, 28] \pmod{41}$$

**Example 8.23** A  $\text{BRGDD}_4(5, 2^6)$  exists. Let  $\{i, i + 6\}$  be the groups.

$$[-0, 1, 2, 4, 9]; [0, 1, -2, -4, 9] \pmod{12}$$

**Example 8.24** A  $\text{BRGDD}_2(5, 4^6)$  exists. Points agreeing in their first two elements are in the same group. (Design adapted from [86, III.1.37]).

$$[000, 012, -020, -101, -112]; [000, 010, -021, 100, 113];$$
$$[000, 011, 102, 112, -121]; [000, 013, -103, -111, 123] \pmod{(-, 3, 4)}$$

# Chapter 9

## Bhaskar Rao Designs of Block Size 6

All the material in this chapter is due to the author of this thesis.

### 9.1 Some Constructions of $BRD(v, 6, \lambda)$

In Chapter 8, It was shown that  $BRD(v, 5, \lambda)$  exists for  $\lambda = 4$  if  $v > 215$ , with 10 smaller possible exceptions and one definite exception at  $v = 5$ ; for  $\lambda = 10$  if  $v > 445$ , with 11 smaller possible exceptions, and one definite exception at  $v = 5$ ; and for  $\lambda = 20$ , with the possible exception of  $v = 32$ . In this chapter, necessary conditions for the existence of Bhaskar Rao designs of block size 6 are given. Some constructions of  $BRD(v, 6, \lambda)$  are also given. Following example gives a BRD of block size 6.

**Example 9.1** There exists a  $BRD(7, 6, 20)$  constructed from  $4 \times BIBD(7, 6, 5)$ :

0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	1	0	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$
1	1	0	1	1	1	1	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	$\bar{1}$	$\bar{1}$	1	$\bar{1}$	$\bar{1}$	0	1	1	1	1	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	$\bar{1}$	$\bar{1}$	
$\bar{1}$	1	1	0	1	1	1	$\bar{1}$	1	$\bar{1}$	0	$\bar{1}$	1	$\bar{1}$	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	0	1	1	1	
1	$\bar{1}$	1	1	0	1	1	1	$\bar{1}$	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	$\bar{1}$	1	$\bar{1}$	1	$\bar{1}$	0	$\bar{1}$	1	1	$\bar{1}$	1	1	0	1	1
$\bar{1}$	$\bar{1}$	$\bar{1}$	1	1	0	1	1	1	$\bar{1}$	$\bar{1}$	$\bar{1}$	0	1	$\bar{1}$	1	1	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	$\bar{1}$	1	1	1	$\bar{1}$	0	$\bar{1}$
$\bar{1}$	$\bar{1}$	$\bar{1}$	1	1	1	0	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	1	$\bar{1}$	0	1	1	$\bar{1}$	1	0	1	1	$\bar{1}$	$\bar{1}$	1	1	1	0	

An alternative representation of this design is presented where the base blocks are developed over  $Z_7$ :

Base block	Signed shifts
(1, 2, 3, 4, 5, 6)	52, 28, 12, 0, 0, 0
(1, 2, 3, 4, 5, 6)	38, 21, 31, 39, 49, 41, 24
(1, 2, 3, 4, 5, 6)	22, 7, 41, 19, 39, 50, 42
(1, 2, 3, 4, 5, 6)	22, 7, 41, 52, 25, 12, 38

Now count the shifts, and the block positions, from zero. The signed shift pattern is given as a binary representation: consider the last block of this design, which is shift 6, and pattern 38. 38 has non-zero bits in position 1, 2 and 5 (again counting from zero), so the shifted base block of  $(0, 1, 2, 3, 4, 5)$  has a signed form of  $(0, -1, -2, 3, 4, -5)$ .

**Theorem 9.1.1** *For the existence of a  $BRD(v, b, r, 6, \lambda)$ , the necessary conditions are:*

1.  $\lambda(v-1) \equiv 0 \pmod{5}$
2.  $\lambda v(v-1) \equiv 0 \pmod{30}$
3.  $\lambda \equiv 0 \pmod{2}$
4.  $b \equiv 0 \pmod{2}$

Condition 4 is implied by condition 3 for  $k = 6$ . Also, if  $v = 6$  it is necessary that  $\lambda \equiv 0 \pmod{4}$  and  $\lambda \geq 8$  (noting Theorem 8.3).

For  $BRD(v, 6, \lambda)$ , these conditions imply the following:

**Table 9.1**

$\lambda$	condition on $v$
2	$v \equiv 1, 6 \pmod{15}$ , with $v > 16$ except $v = 21$
6	$v \equiv 1 \pmod{5}$ , with $v > 11$
10	$v \equiv 0, 1 \pmod{3}$ , with $v > 7$
30	all $v > 7$

**Remark 9.1.2** *A  $BRD(6, 6, \lambda; Z_2)$  can be formed for  $\lambda = 8$  and  $\lambda = 12$  by taking the first 6 rows of a Hadamard matrix of order  $\lambda$  and by adjoining suitable combinations of these two designs, we can form any  $BRD(6, 6, \lambda; Z_2)$  with  $\lambda \equiv 0 \pmod{4}$  and  $\lambda \geq 8$ .*

**Example 9.2** There exists a  $BRD(16, 32, 12, 6, 4)$  constructed from two copies of the  $BIBD(16, 16, 6, 6, 2)$ :

```

11111100000000001111110000000000
11000011110000001100001111000000
10100010001110001010001000111000
10010001001001101001000100100110
1000100010010101000100010010101
1000010001001011000010001001011
0110001000000111010001000000111

```



01010001000110010101000100011001  
 0100100010101010100100100010101010  
 01000100011101000100010001110100  
 00110000111000010011000011100001  
 00101001010100100010100101010010  
 00100101100011000010010110001100  
 00011010010011000001101001001100  
 00010110100100100001011010010010  
 00001111001000010000111100100001

**Remark 9.1.3** No  $BRD(16, 6, 2; Z_2)$  exists, since we could then form a  $BRD(10, 4, 2; Z_2)$  by residualization and this later design was shown to be impossible [42].

**Example 9.3** There exists a  $BRD(12, 6, 10)$  consisting of the following base blocks:  
 $[-0, 1, 3, 7, 8, 10]$ ;  $[0, -1, 3, -7, 8, 10]$ ;  $[\infty, 0, -5, -6, 8, 10]$ ;  $[\infty, -0, 5, 6, -8, 10]$

**Example 9.4** There exists a  $BRD(13, 52, 24, 6, 10)$  constructed from a  $2 \times BIBD(13, 26, 12, 6, 5)$ :

1010001100101101000100011110100011001011010001000111  
 1101000110010110100010001111010001100101101000100011  
 0110100011001111010001000101101000110011110100010001  
 1011010001100111101000100010110100011001111010001000  
 0101101000110011110100010001011010001100111101000100  
 0010110100011001111010001000101101000110011110100010  
 1001011010001000111101000110010110100010001111010001  
 1100101101000100011110100011001011010001000111101000  
 0110010110100010001111010001100101101000100011110100  
 0011001011010001000111101000110010110100010001111010  
 0001100101101000100011110100011001011010001000111101  
 1000110010110100010001111010001100101101000100011110  
 0100011001011010001000111101000110010110100010001111

**Example 9.5** There exists a  $BRD(15, 70, 28, 6, 10)$  constructed from a  $2 \times BIBD(15, 35, 14, 6, 5)$ :

```

1111111111111100000000000000000000000000111111111111110000000000000000000000
1100000101010000111100100001000101110000001010100001111001000010001011
0110000010101010011110010000100010101000000101010100111100100001000101
0011000001010101001111001000110001000100000010101010011110010001100010
1001100000101000100111100100011000110010000001010001001111001000110001
010011000001010001001111001010101000010010000001010001001111001010101000
10100110000010000010011110010101010010100100000100000100111100101010100
0101001100000110000100111100001011001010010000011000010011110000101010
10101001100000010000100111100001011010100100000010000100111100001011
0101010011000000100001001111000010101010010000001000010011111000101
0010101001100010010000100111100010001010100100100100100100001001111100010
00010101001100110010000100110100010001010100100100100100100010011010001
000010101001101110010000100110110000000101010011011001000010011011000
00000101010011111001000010001011000000010101001111100100001000101100
1000001010100101111001000010001011010000010101001111100100001000101100

```

**Example 9.6** There exists a  $BRD(8, 56, 42, 6, 30)$  constructed from a  $2 \times BIBD(8, 28, 21, 6, 15)$ :

```

01101111001111110111111001111011011111010101111101110111
10110111100111111001111111011101011111101010111110111011
11011011110011111001111110111100101111111010111111011101
11101101111001111100111110111110101111110101111110111101
111101101111001111100111101110111110101111111101011110111
011110111111100111110011111011111010111111010111110111011
1011110111111100111110011110111110101011111011110111011
1101111001111111011111001111011111101010111110111011101

```

**Example 9.7** There exists a  $BRD(11, 44, 24, 6, 12)$  constructed from a  $2 \times BIBD(11, 21, 12, 6, 6)$ :

```

10100011101101000111011010001110110100011101
1101000111011010001110-101000--10-101000--10
01101000111011010001--0--010001--0--01000---
1011010001--0-10100011101-010001-10--01000--
110110100011-0--0-0001--0--0-0001--0--0-000-
1110110-000-1-01-0-00011-0--01000-110--01000
011101-0-000-110110-000--10110-000-110--0100
0011101-0-000-1-01-0-0001110--0-0001-101-010
0001110110-000-110--0-000-1-0-1010001-10-101
100011-0--010001--0-10-000-1101-010001110-10
010001110--0-000--10--010001-101-0-000-1-0-1

```

$BRD(21, 6, 12)$ ,  $BRD(18, 6, 20)$ ,  $BRD(19, 6, 20)$ ,  $BRD(21, 6, 20)$  and  $BRD(22, 6, 20)$  were also found using a computer search on BIBDs. These  $BRDs$  have not been listed in this thesis because of their large size.

## 9.2 Conclusion

The results given in this chapter can be used to find general theorems for  $BRD(v, 6, \lambda; Z_2)$ . We leave this as an unsolved problem.

# Chapter 10

---

## Concluding Remarks and Future Work

This chapter concludes this thesis by summarising its aims and by giving an indication as to how successful, in trying to achieving them, the author feels he has been.

The first aim of this thesis was to study minimal structures of combinatorial designs, especially Room squares and latin squares. The uniquely completable and critical sets in Room squares have been studied. General constructions for uniquely completable sets of Room squares have been given but results on minimal and maximal critical sets are empirical only because of the structure of Room squares. General constructions of uniquely completable and critical sets for five different types of back-circulant latin squares have also been given. The terms nest, power, influence and strong box have been introduced and investigated in critical sets of Room squares and back-circulant latin squares. Latin interchanges have been used to construct critical sets in modified-two back-circulant latin squares containing odd order subsquares. Generally problems in latin squares have been found much easier to generalise than Room squares. It has been noted that secret sharing schemes based on Room squares are more efficient than latin squares but very much harder to find. Following are some possible directions for further research in this area:

- The algorithms in Chapter 2 used exhaustive search and hill-climbing techniques to look for minimal and maximal critical sets in Room squares. The algorithms are not efficient for higher orders of Room squares. The algorithms need to be optimised or more efficient algorithms invented.
- Establish theoretical bounds for minimal and maximal critical sets of Room squares.
- Algorithms to construct infinite families of critical sets as well as the smallest and largest critical sets of Room squares need to be devised.

- The first two scenarios, listed in Section 2.1 of critical sets of Room squares have been studied in this thesis. It is noted that results obtained from scenario 2 are much better than those by scenario 1. It is worth investigating other scenarios as well.
- The results about power and influence of entries in critical sets of Room squares are empirical only. Further research is needed to generalise these results and construct hierarchical structures of influences in Room squares. We have also noted that the strong box in Room squares is almost double the size of the strong box in back-circulant latin squares. General construction of strong boxes in Room squares is an open problem to investigate.
- Prove the Conjecture 7.10 that the uniquely completable set for  $MBC(2, 2m+1)$ , for all  $m$ , is a critical set.
- The uniquely completable sets of  $MBC(2, 2m)$ ,  $BCFC(2, 2m)$  and  $BCFC(2, 2m+1)$  back-circulant latin squares are given in Appendix B. The question is to prove that these are critical sets.
- There is also scope to investigate nest, power, influence and strong box in direct product of latin squares, modified-two circulant and back-circulant - forward-circulant latin squares.

The second aim was to use the minimal structures of combinatorial designs in cryptographic applications, particularly, secret sharing schemes. Generalised, hierarchical, key management and perfect secret sharing schemes based on critical sets of Room squares have been proposed. It has also been shown how cheating in secret sharing schemes can be detected and prevented using critical sets of Room squares. Since there is not much known about critical sets of Room squares, the implementation of these schemes is limited at the moment. However, the directions for future research are:

- The schemes proposed in Sections 4.2 and 4.3 are computationally secure as the number of Room squares grow exponentially for higher order Room squares. However there is further need to design schemes based on Room squares which are theoretically secure.
- Quantify the security of the schemes more effectively, that is, which critical sets are more secure than others.

- A general procedure that will start with an access structure and result in a Room square is needed.
- Design and develop schemes based on critical sets of direct product of latin squares and modified-two back-circulant latin squares.

The third aim was to study another combinatorial structure, Bhaskar Rao designs (BRDs), which can also be used in cryptographic functions, particularly perfect hashing functions. In this thesis, most of the cases of  $\text{BRD}(v, 5, \lambda)$  for  $\lambda = 4, 10, 20$  have been solved. A few other cases of  $\text{BRD}(v, 5, \lambda)$  have also been solved and are given in Chapter 8. Construction of  $\text{BRD}(v, 5, 2)$ 's,  $\text{BRD}(v, 5, 6)$ 's and some other cases of  $v$  still need to be investigated. A few problems of  $\text{BRD}(v, 6, \lambda)$ 's have also been solved in Chapter 9, but general construction of  $\text{BRD}(v, 6, \lambda)$ 's is still an open problem.

# Bibliography

---

- [1] C.J. Colbourn and J.H. Dinitz, *The CRC Handbook of Combinatorial Designs* CRC Press, Boca Raton, FL, (1996), 203–213.
- [2] R.J.R. Abel, Forty-three balanced incomplete block designs. *J. Comb. Theory, Series A* **65** (1994), 252–267.
- [3] R.J.R. Abel, *On the Existence of Balanced Incomplete Block Designs and Transversal Designs*, Ph.D. Thesis, University of New South Wales, 1995.
- [4] R.J.R. Abel, Difference families, in: *The CRC Handbook of Combinatorial Designs* (eds. C.J. Colbourn and J.H. Dinitz), CRC Press, Boca Raton, FL, (1996), 270–287.
- [5] R.J.R. Abel, *Personal communication*, Nov. 1997.
- [6] R.J.R. Abel, A.E. Brouwer, C.J. Colbourn and J.H. Dinitz, Mutually orthogonal latin squares, in: *The CRC Handbook of Combinatorial Designs* (eds. C.J. Colbourn and J.H. Dinitz), CRC Press, Boca Raton, FL, (1996), 111–142.
- [7] R.J.R. Abel, G. Ge, M. Greig and L. Zhu, Resolvable balanced incomplete block designs with a block size of 5, *J. Stat. Planning and Inference* **95** (2001), 49–65.
- [8] R.J.R. Abel and M. Greig, BIBDs with small block sizes, in: *The CRC Handbook of Combinatorial Designs* (eds. C.J. Colbourn and J.H. Dinitz), CRC Press, Boca Raton, FL, (1996), 41–47.
- [9] R.J.R. Abel and M. Greig, Some new RBIBDs with block size 5 and PBDs with block sizes  $\equiv 1 \pmod{5}$ , *Australasian J. Comb.* **15** (1997), 177–202.
- [10] R.R. Anstice, On a problem in combinatorics. *Cambridge and Dublin Math. J.* **7** (1852), 279–292.

- [11] R.R. Anstice, On a problem in combinatorics (continued). *Cambridge and Dublin Math. J.* **8** (1852), 149–154.
- [12] D.S. Archdeacon, J.H. Dinitz and W.D. Wallis, Sets of pairwise orthogonal one-factorizations of  $K_{10}$ . *Congr. Numer.* **43** (1984), 45–79.
- [13] J. Benaloh and J. Leichter, Generalised secret sharing and monotone functions. in *Lecture Notes in Computer Science 403; Advances in Cryptology: Proceedings of Crypto'88*, S Goldwasser Ed., Santa Barbara, CA, Aug. 1988, Berlin: Springer-Verlag, (1990), 27–35.
- [14] F.E. Bennett, H-D.O.F. Gronau, A.C.H. Ling and R.C. Mullin, PBD-closure, in: *The CRC Handbook of Combinatorial Designs* (eds. C.J. Colbourn and J.H. Dinitz), CRC Press, Boca Raton, FL, (1996), 203–213.
- [15] T. Beth, D. Jungnickel and H. Lenz, *Design Theory*, Cambridge University Press, Cambridge, UK, 1985.
- [16] M. Bhaskar Rao, Group divisible family of PBIB designs, *J. Indian Stat. Assoc.* **4** (1966), 14–28.
- [17] M. Bhaskar Rao, Balanced orthogonal designs and their applications in the construction of some BIB and group divisible designs. *Sankhyā Ser. A* **32** (1970), 439–448.
- [18] G.R. Blakley, Safeguarding cryptographic keys. *Proc. AFIPS 1979 Natl. Computer Conference*, New York, **48**, (1979), 313–317.
- [19] E.F. Brickel, Some ideal secret sharing schemes. *J. Comb. Math. and Comb. Computing*, **9**, (1989), 105–113.
- [20] A.E. Brouwer, A. Schrijver and H. Hanani, Group divisible designs with block size four, *Disc. Math.* **20** (1977), 1–10.
- [21] G. Chaudhry and J. Seberry, On the  $(10, 5, \lambda)$ -family of Bhaskar Rao designs, *Bulletin of the ICA* **23** (1998), 83–87.
- [22] G. Chaudhry and J. Seberry, Minimal and maximal critical sets in Room squares. *7th Australasian Workshop on Combinatorial Algorithms (AWOCA'96)*, Magnetic Island, Australia, July 15-19 (1996), Technical Report 508, July 1996, Dept. of Computer Science, University of Sydney, Australia, 75–86.



- [23] G. Chaudhry and J. Seberry, Secret sharing schemes based on Room squares. *Proceedings of DMTCS'96*, Auckland Newzealand. *Combinatorics, Complexity and Logic*, Springer-Verlag, Singapore (1996), 158–167.
- [24] G. Chaudhry and J. Seberry, Minimal critical set of a Room square of order 7. *Bulletin of the ICA*, **20** (1997), 90.
- [25] G. Chaudhry and J. Seberry, Room squares: critical sets and their bounds. Presented at *3rd International Conference on Combina. Math. and Comb. Comput. (3ICCMCC'97)*, Melbourne, Australia, July 1997.
- [26] G. Chaudhry, H. Ghodosi and J. Seberry, Perfect secret sharing schemes from Room squares, *J. Comb. Math. and Comb. Computing (JCMCC)*, **28** (1998), 55–61.
- [27] G. Chaudhry, M. Greig and J. Seberry, On the  $(v, 5, \lambda)$ -family of Bhaskar Rao designs, *J. Stat. Planning and inference* (to appear).
- [28] G. Chaudhry and J. Seberry, Influence of entries in critical sets of Room squares. *Bulletin of the ICA*, **28** (2000), 67–74.
- [29] G. Chaudhry and J. Seberry, On uniquely completable Sets in Room squares. *Bulletin of the ICA*. (to be revised)
- [30] G. Chaudhry, J. Seberry and R. Peddada, Uniquely completable sets in latin squares with subsquares of half order, *Aust. J. of Combinatorics* (to be revised).
- [31] G. Chaudhry and J. Seberry, Critical sets in modified-two back-circulant latin squares, (in preparation).
- [32] D. Chaum, Computer systems established, maintained and trusted by mutually suspicious groups, Memorandum No. UCB/ERL M179/10, *University of California*, 1979.
- [33] D. Chen and D.R. Stinson, Recent results on combinatorial constructions for threshold schemes. *Aust. J. of Combinatorics*, **1** (1990), 29–48.
- [34] C.J. Colbourn and J.H. Dinitz, *The CRC Handbook of Combinatorial Designs*, CRC Press, NY 1996.
- [35] C.J. Colbourn, The complexity of completing partial latin squares, *Discrete Applied Mathematics* **8** (1984), 25–30.

- [36] C.J. Colbourn, M.J. Colbourn and D.R. Stinson, The computational complexity of recognizing critical sets, *Proc. First Southeast Asian Graph Theory Colloquium*, Lecture Notes in Mathematics, vol. **1073**, Springer-Verlag, Berlin, Heidelberg, New York (1984), 248–253.
- [37] J. Cooper, D. Donovan and J. Seberry, Latin squares and critical sets of minimal size. *Aust. J. Combinatorics* **4** (1991), 113–120.
- [38] J. Cooper, D. Donovan and J. Seberry, Secret sharing schemes arising from Latin squares. *Bulletin of the ICA*, **12** (1994), 33–43.
- [39] J. Cooper, D. Donovan and R. Gower, Critical sets in direct product of back-circulant latin squares. *Util. Math.* **50** (1996), 127–162.
- [40] D. Curran and G.H.J. van Rees, Critical sets in latin squares. in *Proc. 8th Manitoba Conference on Numerical Mathematics and Computing, (Congressus Numerantium XXII)*, Utilitas Mathematica, Winnipag, (1978), 165–168.
- [41] W. de Launey, *(0, G)-Designs and Applications*, Ph.D. thesis, University of Sydney 1987.
- [42] W. de Launey and D.G. Sarvate, Non-existence of certain GBRDs, *Ars Combinatoria* **18** (1983), 5–20.
- [43] W. de Launey, Bhaskar Rao designs, in: *The CRC Handbook of Combinatorial Designs* (eds. C.J. Colbourn and J.H. Dinitz), CRC Press, Boca Raton, FL, (1996), 241–246.
- [44] W. de Launey, D.G. Sarvate and J. Seberry, Generalised Bhaskar Rao designs with block size three over  $Z_4$ , *ARS Combinatoria* **19A** (1985), 273–285.
- [45] W. de Launey and J. Seberry, On Bhaskar Rao designs of block size four, *Proceedings of the Seminar on Combinatorics and Applications*, Indian Stat. Institute (1982), 311–316.
- [46] W. de Launey and J. Seberry, On generalized Bhaskar Rao designs of block size four, *Congr. Numer.* **41** (1984), 229–294.
- [47] D. Deng, M. Greig and P. Osterg, More c-Bhaskar Rao designs with small block size (preprint).

- [48] A. Dey and C.K. Midha, Generalized balanced matrices and their applications, *Utilitas Math.* **10** (1976), 139–149.
- [49] J.H. Dinitz, D.K. Garnick and B.D. Mackay, Non-isomorphic one-factorizations of  $K_{12}$ . *J. Comb. Design*, **2**(4), (1994), 273–285.
- [50] J.H. Dinitz and D.R. Stinson, A hill-climbing algorithm for the construction of one-factorizations and Room squares. *SIAM J. Algeb. Disc. Math.* **8**(3) (1987), 430–438.
- [51] J.H. Dinitz and D.R. Stinson, *Room squares, Contemporary Design Theory: a Collection of Surveys*. John Wiley & Sons, Inc. New York, (1992), 137–204.
- [52] J.H. Dinitz and W.D. Wallis, Four orthogonal one-factorizations on 10 points. *Ann. Disc. Math.* **26** (1985), 143–150.
- [53] D. Donovan, Some interesting constructions for secret sharing schemes. *Aust. J. of Combinatorics*, **9** (1994), 37–65.
- [54] D. Donovan and J. Cooper, Critical sets in back circulant latin squares, *Aequationes Mathematicae*, **52** (1996), 157–179.
- [55] D. Donovan, A. Howse and P. Adams, A discussion of latin interchanges. *J. Comb. Math. and Comb. Computing*, **23**, (1997), 161–182.
- [56] L. Fitina, J. Seberry and G.R. Chaudhry, Back-circulant latin square and the influence of a set. *Aust. J. of Combinatorics*, **20** (1999), 163–180.
- [57] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, 1979.
- [58] P.B. Gibbons, *Computing Techniques for the Construction and Analysis of Block Designs*, PhD Thesis, University of Toronto, Technical Report No. 92, May 1976.
- [59] P.B. Gibbons and R. Mathon, Construction methods for Bhaskar Rao and related designs, *J. Aust. Math. Soc. (Series A)* **42** (1987), 5–30; *ibid.* **43** 420.
- [60] R.A. Gower, *Critical sets in Latin Squares and their Application to Minimal Defining Sets of Designs*. PhD Thesis, University of Queensland, Australia, 1995.
- [61] M. Greig, S.P. Hard and D.G. Sarvate, General constructions of c-Bhaskar Rao designs and the  $(c, \lambda)$  spectrum of a  $c - BRD(v, k, \lambda)$  (preprint).

- 
- [62] M. Greig, S.P. Hard, J.S. McCranie and D.G. Sarvate, On c-Bhaskar Rao designs with block size 4 (preprint).
- [63] K.B. Gross, Equivalence of Room designs I. *J. Comb. Theory A* **16** (1974), 264–265.
- [64] K.B. Gross, Equivalence of Room designs II. *J. Comb. Theory A* **17** (1974), 299–316.
- [65] M. Hall, Jr., *Combinatorial Theory*, (2nd ed.), John Wiley and Sons, New York, 1986.
- [66] H. Hanani, Balanced incomplete block designs and related designs, *Disc. Math.* **11** (1975), 255–369.
- [67] S.P. Hard and D.G. Sarvate, On c-Bhaskar Rao designs, *J. Stat. Planning and inference*, **90** (2000), 161–175.
- [68] S.P. Hard and D.G. Sarvate, All c-Bhaskar Rao designs with block size 3 and  $c \geq -1$  exists, *ARS Combinatoria*, (to appear).
- [69] S.P. Hard and D.G. Sarvate, All c-Bhaskar Rao designs with block size 3 and negative  $c$ . *J. Comb. Math. and Comb. Computing (JCMCC)*, (to appear).
- [70] S.P. Hard and D.G. Sarvate, On c-Hadamard matrices. *ARS Combinatoria*, (to appear).
- [71] G. Hossein, J. Pieprzyk, G.R. Chaudhry and J. Seberry. How to prevent cheating in Pinch's scheme, *Electronics Letters*, **33**(17) (1997), 1453–1454.
- [72] M. Ito, A. Saito and T. Nishizeki, Secret sharing schemes realising general access structure, *Proc. IEEE Global Telecomm. Conf. Globecom'87*, Tokyo (1987), 99–102.
- [73] D. Keedwell, Critical sets and critical partial latin squares, *Proc. Third China-USA International Conference on Graph Theory, Combinatorics, Algorithms and Applications*, Beiging, June 1993.
- [74] A. Khodkar, *On smallest critical sets for the elementary Abelian 2-group*, (manuscript).

- 
- [75] T.P. Kirkman, On a problem in combinations. *Cambridge and Dublin Math. J.* **2** (1847), 191–204.
- [76] W.L. Kocay and D.R. Stinson, On strong starters in cyclic groups. *Disc. Math.* **56** (1985), 45–60.
- [77] E.D. Karnin, J.W. Greene and M.E. Hellman, On secret sharing systems. *IEEE Trans. Inf. Th.* **IT-29** (1983), 35–41.
- [78] C. Lam and J. Seberry, Generalized Bhaskar Rao designs, *J. Stat. Plann. and Inference* **10** (1984), 83–95.
- [79] C.H.A. Ling, *Pairwise Balanced Designs and Related Codes*, Ph.D. thesis, University of Waterloo (1997).
- [80] K.M. Martin, *Discrete Structures in the Theory of Secret Sharing*. PhD thesis, Royal Holloway and Bedford New College, University of London, 1991.
- [81] K.M. Martin, New secret sharing schemes from old. *JCCMCC* **14** (1993), 65–77.
- [82] R. Mathon and A. Rosa, On the  $(15, 5, \lambda)$ -family of BIBDs, *Disc. Math.* **77** (1989), 205–216.
- [83] R. Mathon and A. Rosa,  $2-(v, k, \lambda)$  designs of small order, in: *The CRC Handbook of Combinatorial Designs* (eds. C.J. Colbourn and J.H. Dinitz), CRC Press, Boca Raton, FL, (1996), 3–41.
- [84] R.C. Merkle, *Secrecy, Authentication and Public Key Systems*, PhD Dissertation, Stanford University, 1979.
- [85] R.C. Mullin and W.D. Wallis, The existence of Room squares. *Aequa. Math.* **13** (1975), 1–7.
- [86] R.C. Mullin and H-D.O.F. Gronau, PBDs: recursive constructions, in: *The CRC Handbook of Combinatorial Designs* (eds. C.J. Colbourn and J.H. Dinitz), CRC Press, Boca Raton, FL, (1996), 193–203.
- [87] J. Nelder, Critical sets in latin squares. *CSIRO Div. of Math. and Stats, Newsletter*, **38** 1977.
- [88] A. Nijenhuis and H.S. Wilf, *Combinatorial Algorithms: for Computers and Calculators*. 2nd Ed. Academic Press, New York, 1978.

- 
- [89] W.D. Palmer, Partial generalized Bhaskar Rao designs over Abelian groups, *Australasian J. Comb.* **6** (1992), 257–266.
- [90] W. Palmer and J. Seberry, Bhaskar Rao designs over small groups, *Ars Combinatoria* **26A** (1988), 125–148.
- [91] D.P. Rajkundlia, Some techniques for constructing infinite families of BIBD's, *Disc. Math.* **44** (1983), 61–96.
- [92] G.J.E. Rawlins, *Compared to What: an Introduction to the Analysis of Algorithms*. W. H. Freeman and Company, New York, 1992.
- [93] T.G. Room, A new type of magic square. *Math. Gazette* **39** (1955), 307.
- [94] J. Seberry, Secret sharing and group identification, *Telecom Technical Report*, June 1991.
- [95] J. Seberry, Regular group divisible designs and Bhaskar Rao designs with block size 3, *J. Stat. Plann. and Inference* **10** (1984), 69–82.
- [96] J. Seberry, Generalized Bhaskar Rao designs of block size 3. *J. Stat. Plann. and Inference* **11** (1985), 273–279.
- [97] J. Seberry and A.P. Street, A stongbox secured secret sharing schemes. *Util. Math.* **57** (2000), 147–163.
- [98] A. Shamir, How to share a secret. *Comm. ACM* **22(11)** (1979), 612–613.
- [99] S.J. Singh, Some Bhaskar Rao designs and applications for  $k = 3, \lambda = 2$ , *University of Indore J. Science* **7** (1982), 8–15.
- [100] G.J. Simmons, An introduction to shared secret and/or shared control schemes and their applications. in *Contemporary Cryptology, the Science of Information Integrity*, IEEE Press, Piscataway, (1991), 441–497.
- [101] B. Smetaniuk, On the minimal critical set of a latin square. *Util. Math.* **16** (1979), 97–100.
- [102] R.G. Stanton and R.C. Mullin, Construction of Room squares. *Ann. Math. Statistics* **39** (1968), 1540–1548.
- [103] D.R. Stinson and G.H.J. Van Rees, Some large critical sets. *Congr. Numer.* **34** (1982), 441–456.

- 
- [104] D.R. Stinson and S.A. Vanstone, A combinatorial approach to threshold schemes, *SIAM J. of Disc. Math.* **1**, (1988), 230–236.
- [105] D.R. Stinson, An explication of secret sharing schemes. *Design, Codes and Cryptography*, **2** (1992), 357–390.
- [106] A.P. Street, Defining sets for  $t$ -designs and critical sets for latin squares. *New Zealand J. of Maths.* **21** (1992), 133–144.
- [107] D.J. Street, Bhaskar Rao designs from cyclotomy, *J. Austral. Math. Soc.* **29** (1980), 425–430.
- [108] D.J. Street and C.A. Rodger, Some results on Bhaskar Rao designs, *Comb. Math. VII, Lecture Notes in Math.* **829** (1980), 238–245.
- [109] M. Tompa and H. Woll, How to share a secret with cheaters. *J. of Cryptology*, **1**(2) (1988), 133–138.
- [110] R. Vyas, Some Bhaskar Rao designs and applications for  $k = 3, \lambda = 4$ , *University of Indore J. Science* **7** (1982), 16–25.
- [111] W.D. Wallis, *Combinatorial Designs*. Marcel Dekker, New York, 1988.
- [112] W.D. Wallis, A.P. Street and J.S. Wallis, Combinatorics: Room squares, sum-free sets, Hadamard matrices. *Lect. Notes Math*, Springer-Verlag, Berlin-Heidelberg-New York, **292** 1972.
- [113] W.D. Wallis, On one-factorizations of complete graphs. *J. of Australian Mathematical Society*, **16** (1973), 161–171.
- [114] J. Yin, A.C.H. Ling, C.J. Colbourn and R.J.R. Abel, The existence of uniform 5-GDDs, *J. Comb. Designs* **5** (1997), 275–299.

# Appendix A

## Room Squares

### A.1 Examples of Room squares

#### A.1.1 Room squares $R9_1 \dots R9_6$ listed in Tables 2.1 and 2.2

0 1	–	4 9	3 7	2 8	–	5 6	–	–
8 9	0 2	–	–	–	5 7	3 4	–	1 6
–	5 8	0 3	–	6 9	2 4	–	1 7	–
–	3 6	7 8	0 4	–	1 9	–	2 5	–
–	7 9	–	1 2	0 5	3 8	–	4 6	–
4 5	–	–	–	–	0 6	1 8	3 9	2 7
–	–	2 6	5 9	1 3	–	0 7	–	4 8
6 7	1 4	–	–	–	–	2 9	0 8	3 5
2 3	–	1 5	6 8	4 7	–	–	–	0 9

0 1	–	4 5	8 9	–	–	–	6 7	2 3
–	0 2	7 9	–	–	5 8	4 6	1 3	–
–	–	0 3	–	6 8	1 2	–	5 9	4 7
–	7 8	1 6	0 4	3 9	–	2 5	–	–
–	6 9	–	–	0 5	3 7	–	2 4	1 8
4 8	–	–	3 5	2 7	0 6	1 9	–	–
3 6	1 5	2 8	–	–	4 9	0 7	–	–
2 9	3 4	–	1 7	–	–	–	0 8	5 6
5 7	–	–	2 6	1 4	–	3 8	–	0 9

0 1	8 9	–	–	6 7	–	4 5	2 3	–
5 8	0 2	–	–	–	7 9	1 3	–	4 6
4 7	–	0 3	5 9	–	–	6 8	–	1 2
3 9	–	2 5	0 4	–	–	–	1 6	7 8
–	3 7	6 9	1 8	0 5	2 4	–	–	–
–	–	4 8	2 7	1 9	0 6	–	–	3 5
–	–	–	3 6	2 8	1 5	0 7	4 9	–
–	5 6	1 7	–	3 4	–	2 9	0 8	–
2 6	1 4	–	–	–	3 8	–	5 7	0 9

0 1	–	–	–	4 7	–	3 9	2 6	5 8
–	0 2	8 9	–	–	1 4	5 6	3 7	–
2 5	–	0 3	6 9	–	–	4 8	–	1 7
–	1 8	2 7	0 4	–	5 9	–	–	3 6
3 4	6 7	–	2 8	0 5	–	–	1 9	–
7 9	–	1 5	–	3 8	0 6	–	–	2 4
6 8	–	–	1 3	2 9	–	0 7	4 5	–
–	4 9	–	5 7	1 6	2 3	–	0 8	–
–	3 5	4 6	–	–	7 8	1 2	–	0 9

0 1	–	–	5 7	2 9	–	4 8	–	3 6
3 4	0 2	1 5	6 9	–	7 8	–	–	–
7 9	–	0 3	2 8	1 6	–	–	4 5	–
–	3 5	8 9	0 4	–	–	–	2 6	1 7
6 8	–	2 7	–	0 5	1 4	3 9	–	–
–	4 9	–	–	–	0 6	1 2	3 7	5 8
2 5	–	4 6	–	3 8	–	0 7	1 9	–
–	6 7	–	1 3	–	5 9	–	0 8	2 4
–	1 8	–	–	4 7	2 3	5 6	–	0 9

0 1	5 9	7 8	3 6	–	2 4	–	–	–
5 6	0 2	–	8 9	–	–	1 4	–	3 7
–	–	0 3	–	4 8	5 7	6 9	1 2	–
–	–	–	0 4	1 3	–	5 8	7 9	2 6
3 9	–	–	2 7	0 5	–	–	4 6	1 8
2 8	1 7	4 9	–	–	0 6	–	3 5	–
–	–	1 6	–	2 9	3 8	0 7	–	4 5
–	3 4	2 5	–	6 7	1 9	–	0 8	–
4 7	6 8	–	1 5	–	–	2 3	–	0 9

#### A.1.2 Room squares $R11_1 \dots R11_3$ listed in Table 2.1

12 1	–	9 6	–	–	–	3 5	7 2	11 10	–	8 4
9 5	12 2	–	10 7	–	–	–	4 6	8 3	1 11	–
–	10 6	12 3	–	11 8	–	–	–	5 7	9 4	2 1
3 2	–	11 7	12 4	–	1 9	–	–	–	6 8	10 5
11 6	4 3	–	1 8	12 5	–	2 10	–	–	–	7 9
8 10	1 7	5 4	–	2 9	12 6	–	3 11	–	–	–
–	9 11	2 8	6 5	–	3 10	12 7	–	4 1	–	–
–	–	10 1	3 9	7 6	–	4 11	12 8	–	5 2	–
–	–	–	11 2	4 10	8 7	–	5 1	12 9	–	6 3
7 4	–	–	–	1 3	5 11	9 8	–	6 2	12 10	–
–	8 5	–	–	–	2 4	6 1	10 9	–	7 3	12 11



12 1	9 5	–	3 2	11 6	8 10	–	–	–	7 4	–
–	12 2	10 6	–	4 3	1 7	9 11	–	–	–	8 5
9 6	–	12 3	11 7	–	5 4	2 8	10 1	–	–	–
–	10 7	–	12 4	1 8	–	6 5	3 9	11 2	–	–
–	–	11 8	–	12 5	2 9	–	7 6	4 10	1 3	–
–	–	–	1 9	–	12 6	3 10	–	8 7	5 11	2 4
3 5	–	–	–	2 10	–	12 7	4 11	–	9 8	6 1
7 2	4 6	–	–	–	3 11	–	12 8	5 1	–	10 9
11 10	8 3	5 7	–	–	–	4 1	–	12 9	6 2	–
–	1 11	9 4	6 8	–	–	–	5 2	–	12 10	7 3
8 4	–	2 1	10 5	7 9	–	–	–	6 3	–	12 11

12 1	8 6	–	11 5	7 10	3 4	–	–	–	9 2	–
–	12 2	9 7	–	1 6	8 11	4 5	–	–	–	10 3
11 4	–	12 3	10 8	–	2 7	9 1	5 6	–	–	–
–	1 5	–	12 4	11 9	–	3 8	10 2	6 7	–	–
–	–	2 6	–	12 5	1 10	–	4 9	11 3	7 8	–
–	–	–	3 7	–	12 6	2 11	–	5 10	1 4	8 9
9 10	–	–	–	4 8	–	12 7	3 1	–	6 11	2 5
3 6	10 11	–	–	–	5 9	–	12 8	4 2	–	7 1
8 2	4 7	11 1	–	–	–	6 10	–	12 9	5 3	–
–	9 3	5 8	1 2	–	–	–	7 11	–	12 10	6 4
7 5	–	10 4	6 9	2 3	–	–	–	8 1	–	12 11

A.1.3 Room squares  $R11_4 \dots R11_6$  listed in Table 2.2

0 1	8 6	–	11 5	7 10	3 4	–	–	–	9 2	–
11 4	–	0 3	10 8	–	2 7	9 1	5 6	–	–	–
–	–	–	3 7	–	0 6	2 11	–	5 10	1 4	8 9
–	9 3	5 8	1 2	–	–	–	7 11	–	0 10	6 4
9 10	–	–	–	4 8	–	0 7	3 1	–	6 11	2 5
8 2	4 7	11 1	–	–	–	6 10	–	0 9	5 3	–
–	0 2	9 7	–	1 6	8 11	4 5	–	–	–	10 3
7 5	–	10 4	6 9	2 3	–	–	–	8 1	–	0 11
3 6	10 11	–	–	–	5 9	–	0 8	4 2	–	7 1
–	1 5	–	0 4	11 9	–	3 8	10 2	6 7	–	–
–	–	2 6	–	0 5	1 10	–	4 9	11 3	7 8	–

11 5	–	10 8	0 4	–	3 7	–	–	–	1 2	6 9
–	–	5 6	10 2	4 9	–	3 1	0 8	–	7 11	–
7 10	1 6	–	11 9	0 5	–	4 8	–	–	–	2 3
3 4	8 11	2 7	–	1 10	0 6	–	5 9	–	–	–
0 1	–	11 4	–	–	–	9 10	3 6	8 2	–	7 5
–	10 3	–	–	–	8 9	2 5	7 1	–	6 4	0 11
8 6	0 2	–	1 5	–	–	–	10 11	4 7	9 3	–
–	–	–	6 7	11 3	5 10	–	4 2	0 9	–	8 1
–	9 7	0 3	–	2 6	–	–	–	11 1	5 8	10 4
–	4 5	9 1	3 8	–	2 11	0 7	–	6 10	–	–
9 2	–	–	–	7 8	1 4	6 11	–	5 3	0 10	–

8 4	1 0	10 7	6 9	–	–	–	3 11	–	–	2 5
–	–	8 11	3 5	2 6	–	0 4	–	7 9	1 10	–
–	7 3	–	–	4 11	2 10	1 6	–	–	5 8	0 9
–	10 11	1 5	–	–	6 0	8 2	–	–	9 3	4 7
7 6	–	–	4 10	–	5 9	–	1 8	2 3	0 11	–
2 11	–	4 9	–	0 5	8 7	–	10 6	–	–	1 3
–	8 9	0 2	–	7 1	–	3 10	–	5 11	4 6	–
9 1	2 4	–	–	3 8	–	5 7	–	0 10	–	11 6
0 3	6 5	–	7 11	–	–	–	2 9	1 4	–	8 10
–	–	3 6	8 0	9 10	11 1	–	5 4	–	2 7	–
5 10	–	–	1 2	–	4 3	9 11	7 0	8 6	–	–

A.2 Examples of critical sets

A.2.1 Minimal and maximal critical sets listed in Table 2.1

Example A.1 : A skew Room square of side 7 (R11)

8 1	-	4 5	6 7	-	-	2 3
5 7	8 2	-	-	-	1 3	4 6
-	5 6	8 3	1 2	-	4 7	-
-	3 7	-	8 4	2 6	-	1 5
3 6	1 4	2 7	-	8 5	-	-
2 4	-	-	3 5	1 7	8 6	-
-	-	1 6	-	3 4	2 5	8 7

Minimal and maximal critical sets for this square are :

	-			-	-	
	8 2	-	-	-		
-		8 3	1 2	-		-
-	3 7	-	8 4		-	
			-	8 5	-	-
	-	-		1 7	8 6	-
-	-		-			

8 1	-	4 5	6 7	-	-	
5 7		-	-	-		4,6
-		8 3	1 2	-		-
-		-			-	1 5
3 6			-		-	-
	-	-			8 6	-
-	-		-			

Example A.2 : A symmetrical Room square of side 7

8 1	3 4	6 7	2 5	-	-	-
4 6	1 5	-	-	-	3 7	8 2
2 7	-	1 4	-	8 3	-	5 6
3 5	-	-	1 7	2 4	8 6	-
-	-	8 5	3 6	-	1 2	4 7
-	2 6	-	8 4	5 7	-	1 3
-	8 7	2 3	-	1 6	4 5	-

Minimal and maximal critical sets for this square are :

	3 4		2 5	-	-	-		8 1	3 4	6 7		-	-	-
	1 5	-	-	-	3 7	8 2		4 6	1 5	-	-	-		
	-	1 4	-	8 3	-			2 7	-	1 4	-		-	
	-	-			8 6	-			-	-	1 7			-
-	-			-		4 7		-	-		3 6	-		4 7
-		-			-			-	2 6	-			-	
-			-					-			-			-

Example A.3 : Room Square of side 9 (R9<sub>1</sub>)

10 1	-	4 9	3 7	2 8	-	5 6	-	-
8 9	10 2	-	-	-	5 7	3 4	-	1 6
-	5 8	10 3	-	6 9	2 4	-	1 7	-
-	3 6	7 8	10 4	-	1 9	-	2 5	-
-	7 9	-	1 2	10 5	3 8	-	4 6	-
4 5	-	-	-	-	10 6	1 8	3 9	2 7
-	-	2 6	5 9	1 3	-	10 7	-	4 8
6 7	1 4	-	-	-	-	2 9	10 8	3 5
2 3	-	1 5	6 8	4 7	-	-	-	10 9

Minimal and maximal critical sets for this square are :

-				-		-	-	10 1	-	4 9	3 7	2 8	-		-	-
10 2	-	-	-			-		8 9		-	-	-	5 7	3 4	-	
-	5 8	10 3	-	6 9	2 4	-		-		10 3	-	6 9		-		-
-		7 8	10 4	-	1 9	-		-	3 6	7 8	10 4	-		-		-
-	7 9	-			3 8	-		-		-	1 2	10 5		-		-
-	-	-	-	10 6	1 8			4 5	-	-	-	-		1 8		
-	-		5 9	1 3	-		-	4 8	-	-	2 6		-		-	
-		-	-	-	-				-	-	-	-	-	-		
-				-	-	-	-		-				-	-	-	

A.2.2 Minimal and maximal critical sets listed in Table 2.2

Example A.4 : A Room square of order 7 (R11)

0 1	-	4 5	6 7	-	-	2 3
5 7	0 2	-	-	-	1 3	4 6
-	5 6	0 3	1 2	-	4 7	-
-	3 7	-	0 4	2 6	-	1 5
3 6	1 4	2 7	-	0 5	-	-
2 4	-	-	3 5	1 7	0 6	-
-	-	1 6	-	3 4	2 5	0 7

A small cs of size 10 and a large cs of size 23 :

		4 5					
5 7							4 6
		0 3	1 2				
	3 7			2 6			
			3 5	1 7			
							0 7

	-	4 5		-	-		
			-	-			4 6
-	5 6	0 3		-	4 7	-	
	3 7	-			-		
		2 7	-	0 5			
	-				0 6		
	-						0 7

Example A.5 : A Room square of order 9 ( $R9_1$ )

0 1	-	4 9	3 7	2 8	-	5 6	-	-
8 9	0 2	-	-	-	5 7	3 4	-	1 6
-	5 8	0 3	-	6 9	2 4	-	1 7	-
-	3 6	7 8	0 4	-	1 9	-	2 5	-
-	7 9	-	1 2	0 5	3 8	-	4 6	-
4 5	-	-	-	-	0 6	1 8	3 9	2 7
-	-	2 6	5 9	1 3	-	0 7	-	4 8
6 7	1 4	-	-	-	-	2 9	0 8	3 5
2 3	-	1 5	6 8	4 7	-	-	-	0 9

A small cs of size 19 and a large cs of size 38 :

			3 7	2 8				
8 9					5 7			1 6
	5 8	0 3		6 9	2 4			
			0 4					
4 5						1 8	3 9	
		2 6		1 3				
6 7					2 9	0 8		
		1 5						

			3 7	2 8	-			-
8 9	0 2	-		-		3 4	-	1 6
-					2 4			-
						-		-
-	7 9	-	1 2				4 6	-
	-				0 6			2 7
		2 6		1 3		0 7	-	4 8
		-		-	-		0 8	
2 3			6 8		-	-		

Example A.6 : A Room square of order 11 ( $R11_2$ )

11 5	-	10 8	0 4	-	3 7	-	-	-	1 2	6 9
-	-	5 6	10 2	4 9	-	3 1	0 8	-	7 11	-
7 10	1 6	-	11 9	0 5	-	4 8	-	-	-	2 3
3 4	8 11	2 7	-	1 10	0 6	-	5 9	-	-	-
0 1	-	11 4	-	-	-	9 10	3 6	8 2	-	7 5
-	10 3	-	-	-	8 9	2 5	7 1	-	6 4	0 11
8 6	0 2	-	1 5	-	-	-	10 11	4 7	9 3	-
-	-	-	6 7	11 3	5 10	-	4 2	0 9	-	8 1
-	9 7	0 3	-	2 6	-	-	-	11 1	5 8	10 4
-	4 5	9 1	3 8	-	2 11	0 7	-	6 10	-	-
9 2	-	-	-	7 8	1 4	6 11	-	5 3	0 10	-

A small cs of size 33 and a large cs of size 54 :

11 5		10 8			3 7				1 2	
		5 6				3 1			7 11	
	1 6		11 9	0 5		4 8				
		2 7								
0 1		11 4					3 6	8 2		
					8 9		7 1			
8 6	0 2						10 11			
			6 7	11 3	5 10			0 9		
									5 8	10 4
	4 5					0 7		6 10		
9 2						6 11			0 10	

11 5		10 8			3 7				1 2	6 9
--	--		10 2			3 1	0 8	--		
		--	11 9			4 8	--	--	--	2 3
		2 7		1 10	0 6					
							3 6	8 2	--	7 5
	10 3	--				2 5				
8 6		--					10 11		9 3	
		--		11 3	5 10				--	8 1
						--	--	11 1	5 8	10 4
--		9 1		--			--	--	--	--
		--	--	7 8	1 4		--			--

Example A.7 A Room square of order 13 ( $R13_1$ ) and a critical set of size 56 :

--	10 5	6 1	--	--	--	11 2	8 0	4 9	3 12	--	--	7 13
--	2 6	5 9	11 0	12 4	--	--	--	3 10	8 7	1 13	--	--
--	--	--	--	8 1	3 4	--	9 13	--	10 6	2 7	5 11	0 12
4 5	12 8	--	7 10	9 11	--	3 6	--	1 2	--	--	0 13	--
2 8	0 9	11 13	--	5 3	--	--	1 7	12 6	--	4 10	--	--
--	1 11	--	4 13	--	9 12	7 5	--	--	--	0 6	2 10	3 8
9 7	--	4 0	--	10 13	6 8	--	11 3	--	--	--	12 1	5 2
10 1	--	--	6 9	0 2	--	--	--	13 8	4 11	5 12	3 7	--
12 11	--	--	8 5	7 6	13 2	0 10	--	--	--	9 3	--	1 4
--	13 3	--	2 12	--	--	8 4	6 5	7 11	1 0	--	--	9 10
--	--	2 3	--	--	0 7	9 1	10 12	--	5 13	8 11	6 4	--
6 13	--	12 7	1 3	--	10 11	--	2 4	0 5	--	--	8 9	--
0 3	7 4	8 10	--	--	1 5	12 13	--	--	2 9	--	--	6 11

	10 5					11 2	8 0					7 13
	2 6	5 9		12 4			--	3 10				
				8 1	3 4							0 12
	12 8			9 11		3 6		1 2				
	0 9						1 7	12 6				
	1 11				9 12	7 5				0 6	2 10	3 8
9 7		4 0		10 13							12 1	5 2
10 1	--		6 9						4 11	5 12		
12 11				7 6	13 2					9 3		
						8 4	6 5	7 11	1 0			
		2 3	--		0 7	9 1			5 13			--
6 13	--			--			2 4					
		8 10		--			--					

Example A.8 A Room square of order 15 ( $R15_1$ ) and a critical set of size 87 :

2 6	15 9	13 1	--	0 14	10 12	5 3	--	--	7 8	--	4 11	--	--	--
--	3 1	5 14	--	--	--	--	6 15	13 2	0 9	4 10	7 12	11 8	--	--
7 11	--	2 15	0 10	6 8	--	--	5 12	--	1 4	--	--	13 14	3 9	--
3 13	--	7 6	9 12	--	2 1	--	--	--	10 15	0 11	5 8	--	--	4 14
--	8 0	--	--	--	--	14 12	11 13	5 9	--	1 6	3 15	--	4 7	2 10
--	--	--	--	--	15 7	--	0 3	6 4	11 14	12 13	--	9 10	2 8	1 5
15 12	--	--	5 6	1 11	4 3	8 13	--	--	--	7 9	0 2	--	14 10	--
--	--	--	--	--	13 0	7 2	9 14	10 1	6 12	3 8	--	4 5	11 15	--
8 14	6 10	--	1 15	4 12	5 11	--	--	0 7	2 3	--	--	--	--	9 13
0 1	7 14	12 3	4 8	2 9	--	11 10	--	--	--	5 15	--	--	6 13	--
--	--	0 4	--	--	14 6	1 9	7 10	3 11	13 5	--	--	2 12	--	8 15
5 10	4 13	--	11 2	--	8 9	0 6	--	14 15	--	--	--	--	12 1	3 7
--	12 11	10 8	3 14	15 13	--	--	4 2	--	--	--	6 9	1 7	0 5	--
--	--	9 11	--	7 5	--	4 15	1 8	--	--	2 14	10 13	3 6	--	0 12
4 9	2 5	--	7 13	10 3	--	--	--	8 12	--	--	1 14	15 0	--	6 11

			--	0 14	10 12	5 3				--	4 11	--		--
	3 1	5 14					6 15				7 12			
			0 10				5 12		1 4			13 14	3 9	
3 13		7 6					--			0 11	5 8	--		
--	8 0			--		14 12	11 13				3 15			2 10
		--					6 4					9 10	2 8	1 5
15 12	--				4 3	8 13		--					14 10	
					13 0			10 1	6 12		--	4 5	11 15	
8 14			1 15	4 12				0 7	2 3					
0 1	7 14			2 9					--	5 15				
						1 9	7 10	3 11	13 5		--			
	4 13	--	11 2		8 9			14 15		--		--	12 1	
--		10 8		15 13			4 2	--	--		6 9			
		9 11	--	7 5	--	4 15					10 13			
	2 5		7 13	10 3						--		15 0		

Example A.9 A Room square of order 17 ( $R17_3$ ) and a critical set of size 109 :

10 15	17 6	--	--	0 4	13 5	--	--	3 16	--	9 8	2 12	--
4 9	--	--	--	1 15	--	--	--	7 13	6 11	2 3	0 16	--
11 2	13 14	--	--	--	1 3	--	17 9	15 8	--	--	--	5 10
5 7	10 0	1 8	2 13	--	12 14	--	--	--	--	6 15	--	3 9
--	3 4	6 9	--	--	--	--	7 15	--	16 10	--	17 8	0 11
--	--	11 13	3 10	2 7	--	14 16	--	--	--	0 5	1 9	--
--	2 9	--	15 4	--	8 11	12 17	--	--	0 13	7 10	5 3	6 14
--	5 1	--	--	6 8	17 16	0 2	--	--	12 15	--	10 14	--
3 17	8 7	14 15	5 11	16 9	--	--	--	4 12	--	--	--	--
--	--	5 16	12 9	--	--	11 7	--	1 10	2 8	4 14	--	17 13
0 6	--	7 12	16 8	3 13	--	1 4	5 14	--	--	11 17	--	2 15
--	12 11	10 17	6 1	--	7 0	3 15	13 4	--	9 14	--	--	--
1 12	--	--	--	5 17	4 10	8 13	6 16	2 14	3 7	--	--	--
--	--	--	--	12 10	--	--	8 0	11 9	--	1 13	6 7	4 16
8 14	16 15	--	--	--	2 6	5 9	12 3	0 17	--	--	4 11	1 7
13 16	--	0 3	7 17	11 14	9 15	6 10	1 2	--	4 5	--	--	8 12
--	--	2 4	0 14	--	--	--	11 10	5 6	1 17	16 12	15 13	--

11 1	--	7 14	--
17 14	5 12	--	8 10
0 12	--	6 4	7 16
--	16 11	--	4 17
--	14 1	12 13	2 5
8 4	--	15 17	6 12
--	--	1 16	--
--	4 7	11 3	9 13
--	6 13	10 2	1 0
6 3	0 15	--	--
--	9 10	--	--
2 16	--	8 5	--
--	--	9 0	15 11
5 15	17 2	--	3 14
10 13	--	--	--
--	--	--	--
7 9	3 8	--	--

	17 6				13 5			3 16		9 8	2 12					
				1 15								--	17 14	5 12		8 10
11 2						--	17 9	15 8	--	--					6 4	7 16
5 7	10 0													16 11		4 17
	3 4										17 8	0 11			12 13	2 5
		11 13	3 10	2 7		14 16	--				1 9	--			15 17	
	2 9		15 4		8 11					0 13	7 10	5 3		--		
--	5 1			6 8	17 16	0 2				12 15				4 7	11 3	
	8 7	14 15		16 9				4 12						6 13	10 2	
--			12 9			11 7				2 8	4 14	--		6 3	0 15	--
0 6	--			3 13									2 15	9 10		
		10 17	6 1		7 0		13 4			9 14				2 16		
1 12				5 17		8 13	6 16	2 14						9 0		
				12 10				11 9					4 16	5 15		3 14
	16 15	--	--			5 9	12 3			--	4 11	1 7				
		0 3	7 17			6 10				--				--		
--			0 14		--	11 10	5 6	16 12			7 9	3 8				

Example A.10 A Room square of order 19 ( $R19_1$ ) and a critical set of size 130 :

7 2	--	--	--	10 11	14 13	0 4	12 16	--	--	--	6 3	19 5
--	7 16	12 11	--	--	2 5	--	3 0	--	9 13	--	1 10	--
16 18	--	6 15	17 14	--	3 4	7 9	10 13	--	--	--	--	--
--	3 14	--	--	4 5	0 1	--	2 8	19 6	--	12 17	15 16	--
19 8	15 2	--	--	6 18	11 16	--	--	5 10	--	3 13	--	1 12
--	19 9	--	--	--	6 8	--	7 11	--	3 5	1 15	14 18	4 17
5 1	--	19 13	4 8	--	--	10 17	--	18 0	16 2	7 6	--	--
--	13 6	4 1	--	12 15	--	--	19 14	9 2	--	--	17 7	--
0 12	10 18	--	5 9	--	--	--	6 17	--	--	--	--	--
--	17 11	10 14	12 7	3 19	--	--	--	--	1 18	8 9	--	6 16
--	0 5	--	6 11	1 7	17 18	13 2	--	12 8	15 14	--	--	9 10
17 3	--	--	--	8 13	--	--	--	7 15	4 11	0 19	--	--
9 15	12 4	5 16	0 10	--	--	1 6	--	13 17	--	11 18	19 2	3 8
--	--	3 18	19 1	--	10 15	5 12	--	4 14	--	--	0 8	2 11
4 10	--	--	18 15	--	--	19 16	9 1	3 11	7 0	14 5	12 13	--
--	--	--	13 16	9 17	7 19	8 18	--	--	12 6	10 2	5 11	0 14
--	--	7 8	3 2	--	--	14 11	--	16 1	--	--	4 9	13 15
6 14	--	2 17	--	16 0	9 12	15 3	18 4	--	10 8	--	--	--
11 13	8 1	9 0	--	2 14	--	--	15 5	--	19 17	16 4	--	7 18

1 17	--	8 15	18 9	--	--
18 19	8 14	--	17 15	--	4 6
8 5	--	12 19	11 1	--	0 2
--	7 10	--	--	13 18	11 9
--	17 0	4 7	--	9 14	--
12 2	--	--	0 13	--	10 16
11 15	--	3 9	12 14	--	--
3 10	--	11 0	16 8	--	5 18
4 13	2 1	14 16	3 7	8 11	15 19
--	5 13	--	2 4	0 15	--
--	3 16	--	--	4 19	--
9 16	--	2 18	6 5	12 10	1 14
7 14	--	--	--	--	--
--	6 9	--	--	16 17	7 13
--	--	--	--	2 6	8 17
--	4 15	--	--	1 3	--
6 0	12 18	5 17	10 19	--	--
--	11 19	13 1	--	5 7	--
--	--	10 6	--	--	3 12

--	--	--	--	10 11	--	--	12 16	--	--	--	6 3	--
--	--	12 11	--	--	2 5	--	3 0	--	--	--	--	--
16 18	--	6 15	--	--	--	7 9	10 13	--	--	--	--	--
--	3 14	--	--	4 5	--	--	2 8	19 6	--	12 17	15 16	--
19 8	15 2	--	--	--	11 16	--	--	--	--	3 13	--	1 12
--	--	--	--	--	6 8	--	--	--	3 5	1 15	14 18	4 17
--	--	19 13	--	--	--	10 17	--	18 0	--	7 6	--	--
--	--	4 1	--	--	--	--	--	9 2	--	--	17 7	--
--	--	--	5 9	--	--	--	--	--	--	--	--	--
--	17 11	--	12 7	--	--	--	--	--	1 18	--	--	--
--	0 5	--	--	--	--	13 2	--	--	15 14	--	--	--
--	--	--	--	8 13	--	--	--	7 15	4 11	0 19	--	--
--	--	5 16	0 10	--	--	1 6	--	13 17	--	--	19 2	3 8
--	--	--	19 1	--	--	--	--	4 14	--	--	--	--
--	--	--	18 15	--	--	--	9 1	--	7 0	14 5	12 13	--
--	--	--	13 16	--	7 19	--	--	--	12 6	--	5 11	0 14
--	--	7 8	3 2	--	--	14 11	--	16 1	--	--	4 9	--
6 14	--	--	--	16 0	9 12	15 3	18 4	--	10 8	--	--	--
--	8 1	9 0	--	2 14	--	--	15 5	--	--	--	--	7 18



1 17		8 15	18 9		
18 19			17 15		
8 5			11 1		
			-		11 9
	17 0	4 7			
		3 9			
		11 0	16 8		5 18
	2 1			8 11	
	5 13	-	2 4	0 15	
				4 19	
9 16		2 18	6 5	12 10	1 14
7 14					
			-	16 17	7 13
	4 15			1 3	
6 0	12 18				
	11 19	13 1			
		10 6			3 12

Example A.11 A Room square of order 21 ( $R21_1$ ) and a critical set of size 165 :

19 18	1 11	-	3 20	-	-	21 10	-	-	-	0 5	-	17 16
-	19 10	14 21	0 16	5 2	12 1	-	-	-	13 18	-	-	9 15
-	15 7	-	8 18	4 14	-	-	-	12 0	1 10	-	2 17	3 13
-	-	10 13	2 19	0 7	20 4	-	21 6	15 11	3 9	-	-	-
-	-	3 7	4 15	-	-	20 9	18 2	-	-	1 14	5 12	-
-	-	-	1 13	9 18	-	8 17	-	14 19	21 4	15 16	-	20 2
6 10	-	-	21 5	8 1	3 15	16 11	-	-	-	7 20	-	12 19
2 8	12 13	5 17	-	-	-	-	11 0	-	-	6 18	3 21	4 10
4 13	21 17	0 1	12 14	15 6	2 11	-	16 9	7 5	-	-	-	-
-	5 20	-	-	11 10	-	7 6	-	16 18	-	3 19	9 1	-
12 20	4 16	6 8	-	19 13	-	-	-	10 2	14 17	-	15 0	-
0 9	-	-	-	-	-	4 12	1 5	21 13	8 15	-	10 16	-
-	-	20 11	6 17	3 16	9 21	-	4 8	-	0 19	-	7 14	-
1 3	-	2 16	-	21 20	5 19	14 15	7 17	4 6	11 12	9 13	-	8 0
15 21	14 2	19 4	-	-	8 16	5 13	12 3	-	-	10 17	-	6 1
14 5	3 8	15 18	-	17 12	10 0	-	-	-	-	-	13 11	-
11 17	-	9 12	-	-	6 14	-	-	20 1	2 7	-	-	5 18
-	6 9	-	-	-	17 18	0 2	15 19	-	16 20	8 11	-	7 21
-	-	-	7 11	-	-	1 19	-	8 9	-	12 21	4 18	-
7 16	-	-	9 10	-	-	18 3	13 20	-	-	-	6 19	11 14
-	0 18	-	-	-	7 13	-	10 14	3 17	5 6	2 4	8 20	-

2 15	4 9	8 14	6 13	-	-	7 12	-
-	17 20	-	11 4	8 7	3 6	-	-
-	-	6 20	-	5 16	9 19	-	21 11
-	-	-	16 14	-	5 8	1 17	12 18
11 19	6 0	16 21	-	13 17	-	10 8	-
6 12	5 11	-	-	-	-	0 3	7 10
9 17	14 18	-	-	-	0 4	-	13 2
1 16	7 19	-	-	15 20	-	-	14 9
20 18	-	-	10 3	-	-	-	8 19
13 8	-	-	12 15	14 0	21 2	-	4 17
-	-	3 11	5 9	-	1 7	18 21	-
-	3 2	17 19	7 18	6 11	-	20 14	-
5 10	-	2 12	-	1 18	-	13 15	-
-	-	10 18	-	-	-	-	-
-	-	9 7	-	-	11 18	-	20 0
7 4	1 21	-	19 20	2 9	-	-	16 6
-	10 15	13 0	21 8	3 4	-	16 19	-
-	-	4 1	-	10 12	13 14	-	3 5
14 3	13 16	5 15	0 17	-	10 20	6 2	-
0 21	8 12	-	1 2	-	17 15	4 5	-
-	-	-	-	21 19	16 12	9 11	1 15

19 18	1 11					21 10				0 5		
		14 21										
	15 7		8 18			-			1 10			3 13
		10 13		0 7	20 4		21 6	15 11	3 9			
		3 7					18 2			1 14	5 12	
			1 13			8 17			21 4	15 16		20 2
			21 5	8 1								12 19
2 8	12 13	5 17					11 0			6 18		
4 13	21 17	0 1		15 6	2 11		16 9					
	5 20			11 10	-	7 6	-				9 1	
12 20	4 16	6 8		19 13					14 17		15 0	
0 9			-	-	-			21 13	8 15	-	10 16	
		20 11	6 17	3 16	9 21			-				
	-	2 16		21 20		14 15	7 17		11 12			8 0
	14 2	19 4							-	10 17		6 1
14 5	3 8	15 18			10 0						13 11	
11 17								20 1				5 18
				17 18		15 19				8 11		
			7 11			1 19		8 9		12 21		
7 16			9 10			18 3	13 20				6 19	11 14
	0 18					10 14				2 4	8 20	

2 15	4 9		6 13			7 12	
	17 20		11 4	8 7	3 6		
-			-	5 16	9 19	-	
	6 0			13 17			
6 12	5 11						
					0 4	-	13 2
1 16	7 19						14 9
			10 3				8 19
13 8				14 0	21 2		4 17
					1 7		
-			7 18			20 14	-
5 10		2 12		1 18		13 15	
		10 18				-	
		9 7		-	11 18		20 0
	1 21		19 20	2 9			
	10 15	13 0	21 8	3 4		16 19	
		4 1	-	10 12		-	3 5
	13 16	5 15					
	8 12				17 15	4 5	
					16 12	9 11	

## A.3 Completion of a partial Room square

In this section, we try to complete the Room square by deleting one entry at a time from the critical set when all empty cells are already known.

.....  
 We try to complete the Room square by deleting one entry at a time  
 from the following critical set  
 .....

A Room Squares of order 7

0	1	--	--	4	5	6	7	--	--	--	--	2	3
5	7	0	2	--	--	--	--	--	--	1	3	4	6
--	--	5	6	0	3	1	2	--	--	4	7	--	--
--	--	3	7	--	--	0	4	2	6	--	--	1	5
3	6	1	4	2	7	--	--	0	5	--	--	--	--
2	4	--	--	--	--	3	5	1	7	0	6	--	--
--	--	--	--	1	6	--	--	3	4	2	5	0	7

A Critical set of size 9 where all empty positions are known

*	*	--	--	4	5	*	*	--	--	--	--	2	3
*	*	*	*	--	--	--	--	--	--	1	3	4	6
--	--	*	*	0	3	1	2	--	--	4	7	--	--
--	--	*	*	--	--	*	*	2	6	--	--	1	5
*	*	*	*	*	*	--	--	*	*	--	--	--	--
*	*	--	--	--	--	*	*	*	*	*	*	--	--
--	--	--	--	*	*	--	--	*	*	*	*	*	*

Completion without (1,3; 4 5) pair

*	*	-	-	*	*	*	*	-	-	-	-	2	3
*	*	*	*	-	-	-	-	-	-	1	3	4	6
-	-	5	6	0	3	1	2	-	-	4	7	-	-
-	-	*	*	-	-	*	*	2	6	-	-	1	5
*	*	*	*	*	*	-	-	*	*	-	-	-	-
*	*	-	-	-	-	*	*	*	*	0	6	-	-
-	-	-	-	*	*	-	-	*	*	2	5	0	7

The number of known pairs is 33

Completion without (1,7; 2 3) pair

*	*	-	-	4	5	*	*	-	-	-	-	*	*
*	*	*	*	-	-	-	-	-	-	1	3	4	6
-	-	5	6	0	3	1	2	-	-	4	7	-	-
-	-	*	*	-	-	*	*	2	6	-	-	1	5
*	*	*	*	*	*	-	-	*	*	-	-	-	-
*	*	-	-	-	-	*	*	*	*	*	*	-	-
-	-	-	-	*	*	-	-	*	*	*	*	*	*

The number of known pairs is 30

Completion without (2,6; 1 3) pair

*	*	-	-	4	5	*	*	-	-	-	-	2	3
*	*	*	*	-	-	-	-	-	-	*	*	4	6
-	-	5	6	0	3	1	2	-	-	4	7	-	-
-	-	*	*	-	-	*	*	2	6	-	-	1	5
*	*	*	*	2	7	-	-	*	*	-	-	-	-
*	*	-	-	-	-	*	*	*	*	*	*	-	-
-	-	-	-	1	6	-	-	*	*	*	*	0	7

The number of known pairs is 33

Completion without (2,7; 4 6) pair

*	*	-	-	4	5	*	*	-	-	-	-	2	3
*	*	*	*	-	-	-	-	-	-	1	3	*	*
-	-	5	6	0	3	1	2	-	-	4	7	-	-
-	-	*	*	-	-	*	*	2	6	-	-	1	5
*	*	*	*	*	*	-	-	*	*	-	-	-	-
*	*	-	-	-	-	*	*	*	*	*	*	-	-
-	-	-	-	*	*	-	-	*	*	*	*	*	*

The number of known pairs is 30

Completion without (3,3; 0 3) pair

*	*	-	-	4	5	*	*	-	-	-	-	2	3
*	*	*	*	-	-	-	-	-	-	1	3	4	6
-	-	*	*	*	*	1	2	-	-	4	7	-	-
-	-	*	*	-	-	*	*	2	6	-	-	1	5
*	*	*	*	*	*	-	-	*	*	-	-	-	-
*	*	-	-	-	-	*	*	*	*	*	*	-	-
-	-	-	-	*	*	-	-	*	*	*	*	0	7

The number of known pairs is 30

Completion without (3,4; 1 2) pair

*	*	-	-	4	5	*	*	-	-	-	-	2	3
*	*	*	*	-	-	-	-	-	-	1	3	4	6
-	-	*	*	0	3	*	*	-	-	4	7	-	-
-	-	*	*	-	-	*	*	2	6	-	-	1	5
*	*	*	*	*	*	-	-	*	*	-	-	-	-
*	*	-	-	-	-	*	*	*	*	*	*	-	-
-	-	-	-	*	*	-	-	*	*	*	*	0	7

The number of known pairs is 30

Completion without (3,6; 4 7) pair

*	*	-	-	4	5	*	*	-	-	-	-	2	3
*	*	*	*	-	-	-	-	-	-	1	3	4	6
-	-	*	*	0	3	1	2	-	-	*	*	-	-
-	-	*	*	-	-	*	*	2	6	-	-	1	5
*	*	*	*	2	7	-	-	*	*	-	-	-	-
*	*	-	-	-	-	*	*	*	*	*	*	-	-
-	-	-	-	1	6	-	-	*	*	*	*	0	7

The number of known pairs is 32

Completion without (4,5; 2 6) pair

*	*	-	-	4	5	*	*	-	-	-	-	2	3
*	*	*	*	-	-	-	-	-	-	1	3	4	6
-	-	5	6	0	3	1	2	-	-	4	7	-	-
-	-	*	*	-	-	*	*	*	*	-	-	1	5

*	*	*	*	*	*	-	-	*	*	-	-	-	-
*	*	-	-	-	-	*	*	*	*	*	*	-	-
-	-	-	-	*	*	-	-	*	*	*	*	0	7

The number of known pairs is 31

Completion without (4,7; 1 5) pair

*	*	-	-	4	5	*	*	-	-	-	-	2	3
*	*	*	*	-	-	-	-	-	-	1	3	4	6
-	-	5	6	0	3	1	2	-	-	4	7	-	-
-	-	*	*	-	-	*	*	2	6	-	-	*	*
*	*	*	*	*	*	-	-	*	*	-	-	-	-
*	*	-	-	-	-	*	*	*	*	*	*	-	-
-	-	-	-	*	*	-	-	*	*	*	*	*	*

The number of known pairs is 30

## A.4 Completion of a partial Room square

In this section, we try to complete the Room square by substituting all other possible entries one by one except the actual member of the critical set when empty cells are not known in advance. If, for a member there is some substitution that leads to a completion (clearly, to some other Room square), then that member is essential for any critical subset of our set. Using these examples, we want to show how hard it is to prove that any subset of a critical set will have two or more completions.



.....  
 We remove each pair of the following critical set, one at a time, and  
 try to complete the Room square by substituting all other possible  
 pairs one by one.  
 .....

A Room square of order 7

0	1	--	--	4	5	6	7	--	--	--	--	2	3
5	7	0	2	--	--	--	--	--	--	1	3	4	6
--	--	5	6	0	3	1	2	--	--	4	7	--	--
--	--	3	7	--	--	0	4	2	6	--	--	1	5
3	6	1	4	2	7	--	--	0	5	--	--	--	--
2	4	--	--	--	--	3	5	1	7	0	6	--	--
--	--	--	--	1	6	--	--	3	4	2	5	0	7

A critical set of size 10

**	**	**	**	4	5	6	7	**	**	**	**	**	**
**	**	0	2	**	**	**	**	**	**	1	3	**	**
**	**	5	6	**	**	1	2	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	2	7	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	1	7	**	**	**	**
**	**	**	**	**	**	**	**	3	4	**	**	0	7

.....  
 Remove (2,2;0 2) pair and try all possible pairs at (2,2)  
 .....

Completion without (2,2;0 2) pair

**	**	**	**	4	5	6	7	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	1	3	**	**
**	**	5	6	**	**	1	2	--	--	**	**	--	--
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	2	7	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	1	7	**	**	**	**
**	**	--	--	**	**	--	--	3	4	**	**	0	7

The number of known pairs is 13

Completion with (2,2;- -) pair

**	**	**	**	4	5	6	7	**	**	**	**	**	**
**	**	--	--	**	**	**	**	**	**	1	3	**	**
**	**	5	6	**	**	1	2	--	--	**	**	--	--
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	2	7	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	1	7	**	**	**	**
**	**	--	--	**	**	--	--	3	4	**	**	0	7

The number of known pairs is 14

Completion with (2,2;0 4) pair

**	**	**	**	4	5	6	7	**	**	**	**	**	**
**	**	0	4	--	--	--	--	**	**	1	3	**	**
**	**	5	6	**	**	1	2	--	--	**	**	--	--
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	--	--	2	7	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	1	7	**	**	**	**

```

** **  -- --  ** **  -- --  3 4  ** **  0 7
The number of known pairs is 17

```

Completion with (2,2;2 4) pair

```

** **  ** **  4 5  6 7  ** **  ** **  ** **
** **  2 4  ** **  ** **  ** **  1 3  -- --
** **  5 6  ** **  1 2  -- --  ** **  -- --
** **  ** **  ** **  ** **  ** **  ** **
** **  ** **  2 7  ** **  ** **  ** **  ** **
** **  ** **  ** **  ** **  1 7  ** **  ** **
** **  -- --  ** **  -- --  3 4  ** **  0 7
The number of known pairs is 15

```

Completion with (2,2;4 7) pair

```

** **  ** **  4 5  6 7  ** **  ** **  ** **
** **  4 7  ** **  ** **  ** **  1 3  ** **
** **  5 6  ** **  1 2  -- --  ** **  -- --
** **  ** **  ** **  ** **  ** **  ** **
** **  ** **  2 7  ** **  ** **  ** **  ** **
** **  ** **  ** **  ** **  1 7  ** **  ** **
** **  -- --  ** **  -- --  3 4  ** **  0 7
The number of known pairs is 14

```

```

.....
Remove (7,7;0 7) pair and try with possible pairs at (7,7)
.....

```

Completion without (7,7; 0 7) pair

```

** **  -- --  4 5  6 7  -- --  -- --  ** **
** **  0 2  -- --  -- --  -- --  1 3  ** **
** **  5 6  ** **  1 2  -- --  ** **  ** **
** **  3 7  ** **  ** **  ** **  ** **  ** **
** **  1 4  2 7  ** **  ** **  ** **  ** **
** **  -- --  ** **  ** **  1 7  ** **  ** **
** **  -- --  ** **  ** **  3 4  ** **  ** **
The number of known pairs is 20

```

Completion with (7,7; - -) pair

```

** **  -- --  4 5  6 7  -- --  -- --  ** **
** **  0 2  -- --  -- --  -- --  1 3  ** **
** **  5 6  ** **  1 2  -- --  ** **  ** **
** **  3 7  ** **  ** **  ** **  ** **  ** **
** **  1 4  2 7  ** **  ** **  ** **  ** **
** **  -- --  ** **  ** **  1 7  ** **  ** **
** **  -- --  ** **  ** **  3 4  ** **  -- --
The number of known pairs is 21

```

Completion with (7,7; 0 1) pair

```

-- --  -- --  4 5  6 7  -- --  -- --  2 3
** **  0 2  -- --  -- --  -- --  1 3  ** **
** **  5 6  ** **  1 2  -- --  ** **  ** **
** **  3 7  ** **  ** **  ** **  ** **  ** **
** **  1 4  2 7  ** **  ** **  ** **  -- --
** **  -- --  ** **  ** **  1 7  ** **  ** **
** **  -- --  -- --  -- --  3 4  ** **  0 1

```

The number of known pairs is 26

Completion with (7,7; 0 5) pair

0	1	--	--	4	5	6	7	--	--	--	--	2	3
**	**	0	2	--	--	--	--	--	--	1	3	**	**
**	**	5	6	**	**	1	2	--	--	**	**	**	**
**	**	3	7	**	**	**	**	2	5	**	**	**	**
**	**	1	4	2	7	**	**	0	6	--	--	--	--
**	**	--	--	**	**	**	**	1	7	**	**	**	**
**	**	--	--	**	**	--	--	3	4	**	**	0	5

The number of known pairs is 28

Completion with (7,7; 0 6) pair

Although completed but not a Room square

0	1	--	--	4	5	6	7	--	--	--	--	2	3
--	--	0	2	--	--	--	--	--	--	1	3	4	7
--	--	5	6	--	--	1	2	--	--	0	7	--	--
--	--	3	7	--	--	0	4	2	6	--	--	1	5
3	6	1	4	2	7	--	--	0	5	--	--	--	--
2	4	--	--	0	3	--	--	1	7	--	--	--	--
--	--	--	--	--	--	--	--	3	4	2	5	0	6

The number of known pairs is 49

Completion with (7,7; 1 5) pair

**	**	--	--	4	5	6	7	--	--	--	--	**	**
**	**	0	2	--	--	--	--	--	--	1	3	**	**
**	**	5	6	**	**	1	2	--	--	**	**	**	**
**	**	3	7	**	**	**	**	**	**	**	**	**	**
**	**	1	4	2	7	**	**	**	**	**	**	**	**
**	**	--	--	**	**	**	**	1	7	**	**	**	**
**	**	--	--	**	**	--	--	3	4	**	**	1	5

The number of known pairs is 22

Completion with (7,7; 1 6) pair

**	**	--	--	4	5	6	7	--	--	--	--	**	**
**	**	0	2	--	--	--	--	--	--	1	3	**	**
**	**	5	6	**	**	1	2	--	--	**	**	**	**
**	**	3	7	**	**	**	**	**	**	**	**	**	**
**	**	1	4	2	7	**	**	**	**	**	**	**	**
**	**	--	--	**	**	**	**	1	7	**	**	**	**
**	**	--	--	--	--	**	**	3	4	**	**	1	6

The number of known pairs is 22

Completion with (7,7; 2 5) pair

**	**	--	--	4	5	6	7	--	--	--	--	**	**
**	**	0	2	--	--	--	--	--	--	1	3	**	**
**	**	5	6	**	**	1	2	--	--	**	**	**	**
**	**	3	7	**	**	**	**	**	**	**	**	**	**
**	**	1	4	2	7	**	**	**	**	**	**	**	**
**	**	--	--	**	**	**	**	1	7	**	**	**	**
**	**	--	--	**	**	--	--	3	4	**	**	2	5

The number of known pairs is 22

Completion with (7,7; 2 6) pair

```

** **  -- --    4 5    6 7    -- --    -- --    ** **
** **    0 2    -- --    -- --    -- --    1 3    ** **
** **    5 6    ** **    1 2    -- --    ** **    ** **
** **    3 7    ** **    ** **    ** **    ** **    ** **
** **    1 4    2 7    ** **    ** **    ** **    ** **
** **    -- --    ** **    ** **    1 7    ** **    ** **
** **    -- --    ** **    ** **    3 4    ** **    2 6

```

The number of known pairs is 21

Completion with (7,7; 5 7) pair

```

** **  -- --    4 5    6 7    -- --    -- --    ** **
-- --    0 2    -- --    -- --    -- --    1 3    4 6
** **    5 6    ** **    1 2    -- --    ** **    ** **
** **    3 7    ** **    ** **    ** **    ** **    ** **
** **    1 4    2 7    ** **    ** **    ** **    ** **
** **    -- --    ** **    ** **    1 7    ** **    ** **
** **    -- --    ** **    -- --    3 4    ** **    5 7

```

The number of known pairs is 24

.....  
Remove (3,4; 1 2) pair and try all possible pairs at (3,4)  
.....

Completion without (3,4; 1 2) pair

```

** **  -- --    4 5    6 7    -- --    -- --    ** **
5 7    0 2    -- --    -- --    -- --    1 3    4 6
** **    5 6    ** **    ** **    -- --    ** **    ** **
** **    3 7    ** **    ** **    ** **    ** **    ** **
** **    1 4    2 7    ** **    ** **    ** **    ** **
** **    -- --    ** **    ** **    1 7    ** **    ** **
** **    -- --    ** **    ** **    3 4    ** **    0 7

```

The number of known pairs is 22

Completion with (3,4; - -) pair

```

** **  -- --    4 5    6 7    -- --    -- --    ** **
5 7    0 2    -- --    -- --    -- --    1 3    4 6
** **    5 6    ** **    -- --    -- --    ** **    ** **
** **    3 7    ** **    ** **    ** **    ** **    ** **
** **    1 4    2 7    ** **    ** **    ** **    ** **
** **    -- --    ** **    ** **    1 7    ** **    ** **
** **    -- --    ** **    ** **    3 4    ** **    0 7

```

The number of known pairs is 23

Completion with (3,4; 0 1) pair

```

** **  -- --    4 5    6 7    -- --    -- --    ** **
5 7    0 2    -- --    -- --    -- --    1 3    4 6
** **    5 6    -- --    0 1    -- --    ** **    ** **
** **    3 7    ** **    ** **    ** **    ** **    ** **
** **    1 4    2 7    ** **    ** **    ** **    ** **
** **    -- --    ** **    ** **    1 7    ** **    ** **
** **    -- --    ** **    ** **    3 4    ** **    0 7

```

The number of known pairs is 24

Completion with (3,4; 0 3) pair

```

** **  -- --    4 5    6 7    -- --    -- --    ** **

```

** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 16

Completion with (2,6; - -) pair

** **	** **	4 5	6 7	-- --	** **	** **
** **	0 2	** **	** **	-- --	-- --	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 17

Completion with (2,6; 1 4) pair

** **	** **	4 5	6 7	-- --	** **	** **
** **	0 2	** **	** **	-- --	1 4	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 17

Completion with (2,6; 1 5) pair

** **	** **	4 5	6 7	-- --	** **	** **
** **	0 2	** **	-- --	-- --	1 5	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 18

Completion with (2,6; 1 6) pair

** **	** **	4 5	6 7	-- --	** **	** **
** **	0 2	-- --	** **	-- --	1 6	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
-- --	-- --	-- --	-- --	3 4	2 5	0 7

The number of known pairs is 21

Completion with (2,6; 3 5) pair

** **	** **	4 5	6 7	-- --	** **	** **
** **	0 2	** **	-- --	-- --	3 5	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 18

Completion with (2,6; 3 6) pair

```

** **  ** **  4 5  6 7  -- --  ** **  ** **
** **  0 2  -- --  -- --  -- --  3 6  ** **
** **  5 6  ** **  1 2  -- --  ** **  -- --
** **  ** **  ** **  ** **  ** **  ** **  ** **
** **  ** **  2 7  ** **  ** **  ** **  ** **
** **  -- --  ** **  ** **  1 7  ** **  ** **
** **  -- --  ** **  -- --  3 4  ** **  0 7

```

The number of known pairs is 19

Completion with (2,6; 3 7) pair

```

** **  ** **  4 5  6 7  -- --  ** **  ** **
** **  0 2  ** **  -- --  -- --  3 7  ** **
** **  5 6  ** **  1 2  -- --  ** **  -- --
** **  ** **  ** **  ** **  ** **  ** **  ** **
** **  ** **  2 7  ** **  ** **  ** **  ** **
** **  -- --  ** **  ** **  1 7  ** **  ** **
** **  -- --  ** **  -- --  3 4  ** **  0 7

```

The number of known pairs is 18

Completion with (2,6; 4 6) pair

```

** **  ** **  4 5  6 7  -- --  ** **  ** **
** **  0 2  ** **  ** **  -- --  4 6  ** **
** **  5 6  ** **  1 2  -- --  ** **  -- --
** **  ** **  ** **  ** **  ** **  ** **  ** **
** **  ** **  2 7  ** **  ** **  ** **  ** **
** **  -- --  ** **  ** **  1 7  ** **  ** **
** **  -- --  ** **  -- --  3 4  ** **  0 7

```

The number of known pairs is 17

Completion with (2,6; 4 7) pair

```

** **  ** **  4 5  6 7  -- --  ** **  ** **
** **  0 2  ** **  ** **  -- --  4 7  ** **
** **  5 6  ** **  1 2  -- --  ** **  -- --
** **  ** **  ** **  ** **  ** **  ** **  ** **
** **  ** **  2 7  ** **  ** **  ** **  ** **
** **  -- --  ** **  ** **  1 7  ** **  ** **
** **  -- --  ** **  -- --  3 4  ** **  0 7

```

The number of known pairs is 17

Completion with (2,6; 5 7) pair

```

** **  ** **  4 5  6 7  -- --  ** **  ** **
** **  0 2  ** **  -- --  -- --  5 7  ** **
** **  5 6  ** **  1 2  -- --  ** **  -- --
** **  ** **  ** **  ** **  ** **  ** **  ** **
** **  ** **  2 7  ** **  ** **  ** **  ** **
** **  -- --  ** **  ** **  1 7  ** **  ** **
** **  -- --  ** **  -- --  3 4  ** **  0 7

```

The number of known pairs is 18

.....  
Remove (6,5;1 7) pair and try all possible pairs at (6,5)  
.....

Completion without (6,5; 1 7) pair

** **	-- --	4 5	6 7	** **	-- --	** **
** **	0 2	-- --	-- --	** **	1 3	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	** **	** **	** **	** **	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 17

Completion with (6,5; - -) pair

** **	-- --	4 5	6 7	** **	-- --	** **
** **	0 2	-- --	-- --	** **	1 3	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	** **	** **	** **	-- --	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 18

Completion with (6,5; 0 1) pair

-- --	-- --	4 5	6 7	-- --	-- --	2 3
** **	0 2	-- --	-- --	5 7	1 3	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	2 6	** **	** **
** **	** **	2 7	** **	-- --	** **	** **
** **	** **	** **	** **	0 1	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 24

Completion with (6,5; 0 5) pair

0 1	-- --	4 5	6 7	-- --	-- --	2 3
5 7	0 2	-- --	-- --	-- --	1 3	4 6
-- --	5 6	0 3	1 2	-- --	4 7	-- --
-- --	3 7	-- --	0 4	-- --	-- --	1 5
-- --	-- --	2 7	** **	1 6	-- --	-- --
-- --	** **	-- --	-- --	0 5	2 6	-- --
-- --	-- --	-- --	-- --	3 4	-- --	0 7

The number of known pairs is 47

Completion with (6,5; 0 6) pair

0 1	-- --	4 5	6 7	-- --	-- --	2 3
** **	0 2	-- --	-- --	** **	1 3	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	** **	-- --	** **	0 6	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 22

Completion with (6,5; 1 6) pair

0 1	-- --	4 5	6 7	-- --	-- --	2 3
** **	0 2	-- --	-- --	** **	1 3	** **

```

** ** 5 6 ** ** 1 2 -- -- ** ** -- --
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **   ** ** ** ** ** ** 
** ** ** ** 2 7 ** ** ** ** **   ** ** ** ** 
** ** ** **   ** **   ** ** 1 6 ** ** -- --
** ** -- -- -- -- -- -- 3 4 ** ** 0 7

```

The number of known pairs is 23

Completion with (6,5; 2 5) pair

```

** ** -- -- 4 5 6 7 ** ** -- -- ** **
5 7 0 2 -- -- -- -- -- -- 1 3 4 6
** ** 5 6 ** ** 1 2 -- -- ** ** -- --
** ** ** **   ** **   ** ** ** ** ** 
** ** ** ** 2 7 ** ** ** **   ** ** ** 
** ** ** **   ** **   ** ** 2 5 ** ** -- --
** ** -- -- ** ** -- -- 3 4 ** ** 0 7

```

The number of known pairs is 22

Completion with (6,5; 2 6) pair

```

** ** -- -- 4 5 6 7 ** ** -- -- ** **
** ** 0 2 -- -- -- -- ** ** 1 3 ** **
** ** 5 6 ** ** 1 2 -- -- ** ** -- --
** ** ** **   ** **   ** ** ** 
** ** ** ** 2 7 ** ** **   ** ** ** 
** ** ** **   ** **   ** ** 2 6 ** ** ** 
** ** -- -- ** ** -- -- 3 4 ** ** 0 7

```

The number of known pairs is 18

Completion with (6,5; 5 7) pair

```

** ** -- -- 4 5 6 7 ** ** -- -- ** **
-- -- 0 2 -- -- -- -- -- -- 1 3 4 6
** ** 5 6 ** ** 1 2 -- -- ** ** -- --
** ** ** **   ** **   ** ** ** 
** ** ** ** 2 7 ** ** **   ** ** ** 
** ** ** **   ** **   ** ** 5 7 ** ** ** 
** ** -- -- ** ** -- -- 3 4 ** ** 0 7

```

The number of known pairs is 21

.....  
Remove (5,3;2 7) pair and try all possible pairs at (5,3)  
.....

Completion without (5,3; 2 7) pair

```

0 1 -- -- 4 5 6 7 -- -- -- -- 2 3
5 7 0 2 -- -- -- -- -- -- 1 3 4 6
-- -- 5 6 ** ** 1 2 -- -- ** ** -- --
** ** ** **   ** **   ** ** ** 
** ** ** **   ** **   ** ** ** 
** ** -- -- ** **   ** ** 1 7 ** ** -- --
** ** -- -- ** ** -- -- 3 4 ** ** 0 7

```

The number of known pairs is 26

Completion with (5,3; - -) pair

```

0 1 -- -- 4 5 6 7 -- -- -- -- 2 3
5 7 0 2 -- -- -- -- -- -- 1 3 4 6
-- -- 5 6 ** ** 1 2 -- -- ** ** -- --

```



** **	** **	** **	** **	** **	** **	** **
** **	** **	-- --	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	-- --
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 27

Completion with (5,3; 0 1) pair

-- --	-- --	4 5	6 7	-- --	-- --	2 3
5 7	0 2	-- --	-- --	-- --	1 3	4 6
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	1 5
** **	** **	0 1	** **	** **	** **	-- --
** **	-- --	** **	** **	1 7	** **	-- --
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 28

Completion with (5,3; 0 3) pair

0 1	-- --	4 5	6 7	-- --	-- --	2 3
5 7	0 2	-- --	-- --	-- --	1 3	4 6
-- --	5 6	-- --	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	0 3	-- --	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	-- --
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 29

Completion with (5,3; 0 6) pair

0 1	-- --	4 5	6 7	-- --	-- --	2 3
5 7	0 2	-- --	-- --	-- --	1 3	4 6
-- --	5 6	3 7	1 2	-- --	0 4	-- --
-- --	4 7	-- --	0 3	-- --	-- --	1 5
-- --	-- --	0 6	-- --	2 5	-- --	-- --
** **	-- --	-- --	-- --	1 7	-- --	-- --
-- --	-- --	-- --	-- --	3 4	2 6	0 7

The number of known pairs is 48

Completion with (5,3; 1 6) pair

0 1	-- --	4 5	6 7	-- --	-- --	2 3
5 7	0 2	-- --	-- --	-- --	1 3	4 6
-- --	5 6	-- --	1 2	-- --	** **	-- --
-- --	-- --	2 7	-- --	0 6	-- --	1 5
-- --	** **,	1 6	** **	2 5	** **	-- --
-- --	-- --	0 3	-- --	1 7	** **	-- --
2 6	-- --	-- --	-- --	3 4	-- --	0 7

The number of known pairs is 44

Completion with (5,3; 2 3) pair

** **	-- --	4 5	6 7	-- --	-- --	-- --
5 7	0 2	-- --	-- --	-- --	1 3	4 6
** **	5 6	-- --	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 3	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 25

Completion with (5,3; 2 6) pair

0	1	--	--	4	5	6	7	--	--	--	--	2	3
5	7	0	2	--	--	--	--	--	--	1	3	4	6
--	--	5	6	--	--	1	2	--	--	--	--	--	--
--	--	3	7	--	--	0	4	--	--	--	--	1	5
--	--	--	--	2	6	--	--	0	5	4	7	--	--
**	**	--	--	0	3	--	--	1	7	--	--	--	--
--	--	--	--	--	--	--	--	3	4	2	5	0	7

The number of known pairs is 48

Completion with (5,3; 3 6) pair

0	1	--	--	4	5	6	7	--	--	--	--	2	3
5	7	0	2	--	--	--	--	--	--	1	3	4	6
--	--	5	6	--	--	1	2	--	--	**	**	--	--
--	--	**	**	**	**	**	**	**	**	**	**	**	**
--	--	**	**	3	6	**	**	**	**	**	**	**	**
--	--	--	--	--	--	**	**	1	7	**	**	--	--
2	6	--	--	--	--	--	--	3	4	--	--	0	7

The number of known pairs is 35

Completion with (5,3; 3 7) pair

0	1	--	--	4	5	6	7	--	--	--	--	2	3
5	7	0	2	--	--	--	--	--	--	1	3	4	6
--	--	5	6	--	--	1	2	--	--	**	**	--	--
**	**	--	--	**	**	**	**	**	**	**	**	1	5
**	**	1	4	3	7	**	**	**	**	**	**	--	--
**	**	--	--	**	**	**	**	1	7	**	**	--	--
**	**	--	--	**	**	--	--	3	4	**	**	0	7

The number of known pairs is 32

.....  
Remove (7,5;3 4) pair and try all possible pairs at (7,5)  
.....

Completion without (7,5; 3 4) pair

**	**	--	--	4	5	6	7	**	**	--	--	**	**
**	**	0	2	--	--	--	--	**	**	1	3	**	**
**	**	5	6	**	**	1	2	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	2	7	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	1	7	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	0	7

The number of known pairs is 13

Completion with (7,5; - -) pair

**	**	--	--	4	5	6	7	**	**	--	--	**	**
**	**	0	2	--	--	--	--	**	**	1	3	**	**
**	**	5	6	**	**	1	2	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	2	7	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	1	7	**	**	**	**
**	**	**	**	**	**	**	**	--	--	**	**	0	7

The number of known pairs is 14

Completion with (7,5; 2 3) pair

```

** **  -- --  4 5  6 7  -- --  -- --  -- --
** **    0 2  -- --  -- --  ** **    1 3  ** **
** **    5 6  ** **    1 2  ** **    ** **    ** **
** **    ** **    ** **    ** **    ** **    ** **
** **    ** **    2 7  ** **    ** **    ** **    ** **
** **    ** **    ** **    ** **    1 7  ** **    ** **
** **    ** **    ** **    -- --  2 3  ** **    0 7

```

The number of known pairs is 17

Completion with (7,5; 2 4) pair

```

** **  -- --  4 5  6 7  ** **  -- --  ** **
5 7    0 2  -- --  -- --  -- --    1 3  4 6
** **    5 6  ** **    1 2  ** **    ** **    -- --
** **    ** **    ** **    ** **    ** **    ** **
** **    ** **    2 7  ** **    ** **    ** **    ** **
** **    ** **    ** **    ** **    1 7  ** **    ** **
** **    -- --  ** **    ** **    2 4  -- --    0 7

```

The number of known pairs is 20

Completion with (7,5; 2 5) pair

```

** **  -- --  4 5  6 7  ** **  -- --  ** **
** **    0 2  -- --  -- --  ** **    1 3  ** **
** **    5 6  ** **    1 2  ** **    ** **    ** **
** **    ** **    ** **    ** **    ** **    ** **
** **    ** **    2 7  ** **    ** **    ** **    ** **
** **    ** **    ** **    ** **    1 7  ** **    ** **
** **    ** **    ** **    ** **    2 5  ** **    0 7

```

The number of known pairs is 14

Completion with (7,5; 2 6) pair

```

** **  -- --  4 5  6 7  ** **  -- --  ** **
5 7    0 2  -- --  -- --  -- --    1 3  4 6
** **    5 6  ** **    1 2  ** **    ** **    -- --
** **    ** **    ** **    ** **    ** **    ** **
** **    ** **    2 7  ** **    ** **    ** **    ** **
** **    ** **    ** **    ** **    1 7  ** **    ** **
** **    ** **    -- --  ** **    2 6  -- --    0 7

```

The number of known pairs is 20

Completion with (7,5; 3 5) pair

```

0 1  -- --  4 5  6 7  -- --  -- --  2 3
** **    0 2  -- --  -- --  ** **    1 3  ** **
** **    5 6  ** **    1 2  ** **    ** **    -- --
** **    ** **    ** **    ** **    ** **    ** **
** **    ** **    2 7  ** **    ** **    ** **    ** **
** **    ** **    ** **    ** **    1 7  ** **    ** **
** **    ** **    ** **    -- --  3 5  ** **    0 7

```

The number of known pairs is 19

Completion with (7,5; 3 6) pair

```

0 1  -- --  4 5  6 7  -- --  -- --  2 3
5 7    0 2  -- --  -- --  -- --    1 3  4 6

```

```

** **      5 6      ** **      1 2      ** **      ** **      -- --
** **      ** **      ** **      ** **      ** **      ** **      ** **
** **      ** **      2 7      ** **      ** **      ** **      ** **
** **      ** **      ** **      ** **      1 7      ** **      -- --
** **      ** **      -- --      -- --      3 6      ** **      0 7

```

The number of known pairs is 24

Completion with (7,5; 4 6) pair

```

** **      -- --      4 5      6 7      ** **      -- --      ** **
** **      0 2      -- --      -- --      -- --      1 3      -- --
** **      5 6      ** **      1 2      ** **      ** **      ** **
** **      ** **      ** **      ** **      ** **      ** **      ** **
** **      ** **      2 7      ** **      ** **      ** **      ** **
** **      ** **      ** **      ** **      1 7      ** **      ** **
** **      -- --      -- --      ** **      4 6      ** **      0 7

```

The number of known pairs is 18

.....  
 Remove (1,3;4 5) pair and try all possible pairs at (1,3)  
 .....

Completion without (1,3; 4 5) pair

```

** **      ** **      ** **      6 7      ** **      ** **      ** **
** **      0 2      ** **      ** **      -- --      1 3      ** **
** **      5 6      ** **      1 2      -- --      ** **      -- --
** **      ** **      ** **      ** **      ** **      ** **      ** **
** **      ** **      2 7      ** **      ** **      ** **      ** **
** **      -- --      ** **      ** **      1 7      ** **      ** **
** **      -- --      ** **      -- --      3 4      ** **      0 7

```

The number of known pairs is 15

Completion with (1,3; - -) pair

```

** **      ** **      -- --      6 7      ** **      ** **      ** **
** **      0 2      ** **      ** **      -- --      1 3      ** **
** **      5 6      ** **      1 2      -- --      ** **      -- --
** **      ** **      ** **      ** **      ** **      ** **      ** **
** **      ** **      2 7      ** **      ** **      ** **      ** **
** **      -- --      ** **      ** **      1 7      ** **      ** **
** **      -- --      ** **      -- --      3 4      ** **      0 7

```

The number of known pairs is 16

Completion with (1,3; 0 1) pair

```

** **      -- --      0 1      6 7      ** **      ** **      ** **
** **      0 2      ** **      ** **      -- --      1 3      ** **
** **      5 6      -- --      1 2      -- --      ** **      -- --
** **      3 7      ** **      ** **      ** **      ** **      ** **
** **      1 4      2 7      ** **      ** **      ** **      ** **
** **      -- --      ** **      ** **      1 7      ** **      ** **
** **      -- --      -- --      -- --      3 4      ** **      0 7

```

The number of known pairs is 21

Completion with (1,3; 0 3) pair

```

** **      ** **      0 3      6 7      ** **      ** **      ** **
** **      0 2      ** **      ** **      -- --      1 3      ** **
** **      5 6      -- --      1 2      -- --      ** **      -- --

```

** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 17

Completion with (1,3; 0 4) pair

** **	-- --	0 4	6 7	** **	** **	** **
** **	0 2	-- --	** **	-- --	1 3	** **
0 3	5 6	-- --	1 2	-- --	4 7	-- --
** **	3 7	** **	** **	** **	** **	** **
-- --	1 4	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 24

Completion with (1,3; 0 5) pair

** **	** **	0 5	6 7	-- --	** **	** **
** **	0 2	** **	** **	-- --	1 3	** **
** **	5 6	-- --	1 2	-- --	** **	-- --
** **	** **	** **	** **	2 5	** **	** **
** **	** **	2 7	** **	0 6	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 20

Completion with (1,3; 1 4) pair

** **	-- --	1 4	6 7	** **	** **	** **
** **	0 2	-- --	** **	-- --	1 3	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	-- --	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	-- --	-- --	3 4	** **	0 7

The number of known pairs is 20

Completion with (1,3; 1 5) pair

** **	-- --	1 5	6 7	-- --	** **	** **
** **	0 2	** **	** **	-- --	1 3	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	3 7	** **	** **	** **	** **	** **
** **	1 4	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	-- --	-- --	3 4	** **	0 7

The number of known pairs is 21

Completion with (1,3; 3 5) pair

** **	** **	3 5	6 7	-- --	** **	** **
** **	0 2	** **	** **	-- --	1 3	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 17

Completion with (3,2; 3 7) pair

0	1	--	--	4	5	6	7	--	--	--	--	2	3
**	**	0	2	--	--	--	--	**	**	1	3	**	**
**	**	3	7	**	**	1	2	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	2	7	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	1	7	**	**	**	**
**	**	**	**	**	**	--	--	3	4	**	**	0	7

The number of known pairs is 18

Completion with (3,2; 4 6) pair

0	1	--	--	4	5	6	7	--	--	--	--	2	3
**	**	0	2	--	--	--	--	**	**	1	3	**	**
**	**	4	6	**	**	1	2	**	**	**	**	--	--
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	2	7	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	1	7	**	**	**	**
**	**	**	**	**	**	--	--	3	4	**	**	0	7

The number of known pairs is 19

Completion with (3,2; 4 7) pair

0	1	--	--	4	5	6	7	--	--	--	--	2	3
**	**	0	2	--	--	--	--	**	**	1	3	**	**
**	**	4	7	**	**	1	2	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	2	7	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	1	7	**	**	**	**
**	**	**	**	**	**	--	--	3	4	**	**	0	7

The number of known pairs is 18

Completion with (3,2; 5 7) pair

0	1	--	--	4	5	6	7	--	--	--	--	2	3
**	**	0	2	--	--	--	--	**	**	1	3	**	**
**	**	5	7	**	**	1	2	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	2	7	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	1	7	**	**	**	**
**	**	**	**	**	**	--	--	3	4	**	**	0	7

The number of known pairs is 18

.....  
Remove (1,4;6 7) pair and try all possible pairs at (1,4)  
.....

Completion without (1,4;6 7) pair

**	**	**	**	4	5	**	**	**	**	**	**	**	**
**	**	0	2	--	--	**	**	--	--	1	3	**	**
**	**	5	6	**	**	1	2	--	--	**	**	--	--
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	2	7	**	**	**	**	**	**	**	**
**	**	--	--	**	**	**	**	1	7	**	**	**	**
**	**	--	--	**	**	--	--	3	4	**	**	0	7

The number of known pairs is 16

Completion with (1,4;- -) pair

** **	** **	4 5	-- --	** **	** **	** **
** **	0 2	-- --	** **	-- --	1 3	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	** **	** **	** **
** **	** **	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 17

Completion with (1,4;0 3) pair

** **	-- --	4 5	0 3	** **	** **	** **
** **	0 2	-- --	** **	-- --	1 3	** **
** **	5 6	-- --	1 2	-- --	** **	-- --
** **	3 7	** **	** **	** **	** **	** **
** **	1 4	2 7	-- --	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 22

Completion with (1,4;0 6) pair

** **	** **	4 5	0 6	-- --	-- --	** **
** **	0 2	-- --	** **	-- --	1 3	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	** **	** **	** **	2 6	** **	** **
** **	** **	2 7	-- --	0 5	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 22

Completion with (1,4;3 6) pair

** **	-- --	4 5	3 6	-- --	-- --	-- --
** **	0 2	-- --	** **	-- --	1 3	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	3 7	** **	** **	** **	** **	** **
** **	1 4	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 23

Completion with (1,4;3 7) pair

** **	-- --	4 5	3 7	** **	** **	** **
** **	0 2	-- --	** **	-- --	1 3	** **
** **	5 6	** **	1 2	-- --	** **	-- --
** **	-- --	** **	** **	** **	** **	** **
** **	1 4	2 7	** **	** **	** **	** **
** **	-- --	** **	** **	1 7	** **	** **
** **	-- --	** **	-- --	3 4	** **	0 7

The number of known pairs is 20

Completion with (1,4;6 7) pair

0 1	-- --	4 5	6 7	-- --	-- --	2 3
5 7	0 2	-- --	-- --	-- --	1 3	4 6
-- --	5 6	0 3	1 2	-- --	4 7	-- --
-- --	3 7	-- --	0 4	2 6	-- --	1 5

3	6		1	4		2	7	--	--		0	5	--	--	--	--
2	4	--	--		--	--		3	5		1	7	0	6	--	--
--	--	--	--		1	6	--	--			3	4	2	5	0	7

The number of known pairs is 49



# Appendix B

## Latin Squares

### B.1 Examples of critical sets for $BC(2, n)$

For these, we used a computer program to established the set claimed is a critical set by systematically removing each element from the set and carrying out a complete search to show the subset could not be completed uniquely.

We use  $csBC(n, 2)$  to denote a critical set.

1		3			6	11
				1	2	
	3	6				
	6					
	5		1	4		36

csBC(2,3)

1	2	3	4	5	6
2	1	4	3	6	5
3	4	5	6	1	2
4	3	6	5	2	1
5	6	1	2	3	4
6	5	2	1	4	3

BC(3,2)  $\equiv$  BC(2,3)

1	2	3	4			14
2	3	1				
3	1				5	
4			1			
		5			2	36

csCDG(2,3)

	2				6		8	22
2	1	4	3					
3								2
4			5		7			
5					2	3		
			7			4		
	8	1						6
				4		6		64

csBC(4,2)

1	2	3	4	5	6	7	8
2	1	4	3	6	5	8	7
3	4	5	6	7	8	1	2
4	3	6	5	8	7	2	1
5	6	7	8	1	2	3	4
6	5	8	7	2	1	4	3
7	8	1	2	3	4	5	6
8	7	2	1	4	3	6	5

$BC(4,2) \equiv BC(2,4)$

1	2	3	4	5	6			28
2	3	4	1	6				
3	4	1	2					
4	1	2	3					7
5	6			1	2			
6				2				
			7				3	64

csBC(2,4)

		3					9		31
	1	4			5		7		
	4	5		7			10		
				8				2	
			8	9				3	
	5	8					1		
							4		
8			9	2				6	
9	10		2		4	5	6		
							5	8	100

$csBC(5,2) \equiv csBC(2,5)$

1	2	3	4	5	6	7				40
2	3	4	5	1	7					
3	4	5	1	2						
4	5	1	2						8	
5	1	2						8	9	
6	7				1	2				
7					1					
				8					3	
			8	9				3	4	100

csCDG(2,5)

1						7	8	9				57
		4		6		8	7				11	
3		5	6	7				11	12	1		
4			5			10				2	1	
5		7			10		12			2	3	
	5			10		12						
			10							5	6	
	7	10		12		2				3	6	
	10								6		8	
		12			1	4	3	6				
11	12	1			4			7				
			1			6		8	7		9	144

csBC(6,2)

1	2	3	4	5	6	7	8	9				63
2	3	4	5	6	1	8	9					
3	4	5	6	1	2	9						
4	5	6	1	2	3							
5	6	1	2	3	4						10	
6	1	2	3	4	5					10	11	
7	8	9				1	2	3				
8	9					2	3					
9						3						
					10						4	
				10	11					4	5	144

csBC(2,6)

	2			5	6			9		11			14	76
2		4				8			9		11		13	
3		5					10		12				1	
4				8					11	13				
5	6	7			10		12		14	1			3	
		8		10										
7				11		13			2			4		6
8	7	10		9	12				1	4				
				13					4			6		8
10	9	12		14			1	4	3					
		13		1								8		
12	11				1		3	6	5				10	
13		1	2	3	4									12
						6		8	7			9		11
														196

csBC(7,2)

1	2	3	4	5	6	7	8	9	10						79
2	3	4	5	6	7	1	9	10							
3	4	5	6	7	1	2	10								
4	5	6	7	1	2	3									
5	6	7	1	2	3									11	
6	7	1	2	3									11	12	
7	1	2	3									11	12	13	
8	9	10					1	2	3						
9	10						2	3							
10							3								
						11									4
					11	12								4	5
				11	12	13						4	5	6	196

csCDG(2,7)

1	2	3	4	5	6	7	8	9	10	11	12					112
2	3	4	5	6	7	8	1	10	11	12						
3	4	5	6	7	8	1	2	11	12							
4	5	6	7	8	1	2	3	12								
5	6	7	8	1	2	3	4									
6	7	8	1	2	3	4	5								13	
7	8	1	2	3	4	5	6							13	14	
8	1	2	3	4	5	6	7							13	14	15
9	10	11	12					1	2	3	4					
10	11	12						2	3	4						
11	12							3	4							
12								4								
							13									5
						13	14								5	6
				13	14	15							5	6	7	256

csBC(2,8)

1	2	3	4	5	6	7	8	9	10	11	12	13						131
2	3	4	5	6	7	8	9	1	11	12	13							
3	4	5	6	7	8	9	1	2	12	13								
4	5	6	7	8	9	1	2	3	13									
5	6	7	8	9	1	2	3	4										
6	7	8	9	1	2	3	4										14	
7	8	9	1	2	3	4											14	15
8	9	1	2	3	4											14	15	16
9	1	2	3	4											14	15	16	17
10	11	12	13						1	2	3	4						
11	12	13							2	3	4							
12	13								3	4								
13									4									
								14										5
							14	15									5	6
						14	15	16								5	6	7
					14	15	16	17							5	6	7	8

csCDG(2,9)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15						175
2	3	4	5	6	7	8	9	10	1	12	13	14	15							
3	4	5	6	7	8	9	10	1	2	13	14	15								
4	5	6	7	8	9	10	1	2	3	14	15									
5	6	7	8	9	10	1	2	3	4	15										
6	7	8	9	10	1	2	3	4	5											
7	8	9	10	1	2	3	4	5	6											16
8	9	10	1	2	3	4	5	6	7									16	17	
9	10	1	2	3	4	5	6	7	8								16	17	18	
10	1	2	3	4	5	6	7	8	9							16	17	18	19	
11	12	13	14	15						1	2	3	4	5						
12	13	14	15							2	3	4	5							
13	14	15								3	4	5								
14	15									4	5									
15										5										
									16											6
									16	17									6	7
								16	17	18								6	7	8
							16	17	18	19							6	7	8	9
																				400

csBC(2,10)

B.2 Examples of critical sets for  $BCFC(2, n)$

We use  $csBCFC(2, n)$  to denote a critical set.

1		3			6	11
2						
				4		
	5					
	6				2	
	4	5		3		36

csBCFC(2,3)

		3		5	6			21
	3	4		6				
			2		8	5		
						6	7	
			8	1			4	
6	7					2		
7							2	
			7				1	64

csBCFC(2,4)

1	2			5	6	7	8		10	36
			5	1			9	10		
	4			2	8			6		
5	1	2				6	7			
		8	9	10			3		5	
		9	10					3	4	
						4		2		
9			7				5			
	6				2					100

csBCFC(2,5)

1	2				6					11		58
		4	5							12	7	
3					2		10	11	12		8	
4			1			10	11				9	
				3				7	8		10	
6	1	2		4								
		9	10		12					5		
				12					3	4	5	
			12	7	8	5			2		4	
10	11					4	5	6		2	3	
11			8		10	3		5	6			
			9				3		5	6		144

csBCFC(2,6)

1	2	3	4			7	8	9	10					76
2	3	4	5	6			9	10						
3	4	5	6	7	1	2	10							
4	5	6	7	1	2	3								
5	6	7	1	2	3								11	
6	7	1	2	3								11	12	
7	1										11	12	13	
8	9	10							4	5	6	7		
9	10									4	5	6		
10											4	5		
						11	4							
					11	12	3	4						
				11	12	13	2	3	4					196

csBCFC(2,7)

1	2	3	4			7	8	9	10	11					99
2	3	4	5	6			1	10	11						
3	4	5	6	7	8	1	2	11							
4	5	6	7	8	1	2	3								
5	6	7	8	1	2	3	4							12	
6	7	8	1	2	3	4							12	13	
7	8	1	2	3	4								12	13	14
8					5						12	13	14	15	
9	10	11									5	6	7	8	
10	11											5	6	7	
11													5	6	
														5	
							12								
						12	13	4							
					12	13	14	3	4						
				12	13	14	15	2	3	4					256

csBCFC(2,8)

1	2	3	4	5	6				10	11	12	13					129
2	3	4	5	6	7			1	11	12	13						
3	4	5	6	7	8	9	1	2	12	13							
4	5	6	7	8	9	1	2	3	13								
5	6	7	8	9	1	2	3	4									
6	7	8	9	1	2	3	4									14	
7	8	9	1	2	3	4										14	15
8	9	1	2	3	4										14	15	16
9														14	15	16	17
10	11	12	13										5	6	7	8	9
11	12	13												5	6	7	8
12	13														5	6	7
13																5	6
									6								
								14	5	6							
							14	15	4	5	6						
						14	15	16	3	4	5						
					14	15	16	17	2	3	4	5					324

csBCFC(2,9)

A Uniquely Completable Set for the BCFC(2,10)																			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15					170
2	3	4	5	6	7	8	9	10	1	12	13	14	15						
3	4	5	6	7	8	9	10	1	2	13	14	15							
4	5	6	7	8	9	10	1	2	3	14	15								
5	6	7	8	9	10	1	2	3	4	15									
6	7	8	9	10	1	2	3	4	5										
7	8	9	10	1	2	3	4	5	6										16
8	9	10	1	2	3	4	5	6									16	17	
9	10	1	2	3	4	5	6									16	17	18	
10	1	2	3	4	5	6	7								16	17	18	19	
11	12	13	14	15										6	7	8	9	10	
12	13	14	15												6	7	8	9	
13	14	15														6	7	8	
14	15																6	7	
15																		6	
									16	5									
								16	17	4	5								
							16	17	18	3	4	5							
						16	17	18	19	2	3	4	5						400

B.3 Examples of critical sets for  $MBC(2, n)$

We use  $csMBC(2, n)$  to denote a critical set.

1	2				6	11
3				4		
	5				1	
	6		2			
		5		3		36

$csMBC(2,3)$

1	2	3	4			14
2	3	1				
3	1				5	
4			3			
		5		2		36

$csMBC(2,3)$

	2	3					21
	3				8		
	4				8	5	
		2	3				7
		7			2	1	
7			6		1		3
8	5			1	4		64

$csMBC(2,4)$

1	2		4	5			23
2		4	1				
	4	1				6	
4	1		3			6	7
5				4			
			6			3	
		6	7			3	2
							64

$csMBC(2,4)$

1	2			5			8		10	37
2		4				8		10		
		5	1				10		7	
4	5		2		9					
	1		3				7			
	7	8		10			3		1	
					4					
		10		7	3	2		5		
9					2	1		4	3	
		7								100

csMBC(2,5)

1	2	3	4	5	6	7				40
2	3	4	5	1	7					
3	4	5	1	2						
4	5	1	2						8	
5	1	2						8	9	
6	7				5	4				
7					4					
				8					3	
			8	9				3	2	100

csMBC(2,5)

1	2	3		5	6	7	8				55
2	3		5	6	1	8					
3		5	6	1	2						
	5	6	1	2						9	
5	6	1	2		4				9	10	
6	1	2		4				9	10	11	
7	8					6	5				
8						5					
					9					4	
				9	10				4	3	
			9	10	11			4	3	2	144

csMBC(2,6)



1	2	3	4	5	6	7	8	9	10					79
2	3	4	5	6	7	1	9	10						
3	4	5	6	7	1	2	10							
4	5	6	7	1	2	3								
5	6	7	1	2	3								11	
6	7	1	2	3								11	12	
7	1	2	3								11	12	13	
8	9	10					7	6	5					
9	10						6	5						
10							5							
						11								4
					11	12							4	3
				11	12	13					4	3	2	196

csMBC(2,7)

1	2	3	4		6	7	8	9	10	11						100
2	3	4		6	7	8	1	10	11							
3	4		6	7	8	1	2	11								
4		6	7	8	1	2	3									
	6	7	8	1	2	3									12	
6	7	8	1	2	3		5							12	13	
7	8	1	2	3			5						12	13	14	
8	1	2	3		5							12	13	14	15	
9	10	11						8	7	6						
10	11							7	6							
11								6								
							12								5	
						12	13							5	4	
					12	13	14						5	4	3	
				12	13	14	15					5	4	3	2	256

csMBC(2,8)

1	2	3	4	5	6	7	8	9	10	11	12	13						132
2	3	4	5	6	7	8	9	1	11	12	13							
3	4	5	6	7	8	9	1	2	12	13								
4	5	6	7	8	9	1	2	3	13									
5	6	7	8	9	1	2	3	4										
6	7	8	9	1	2	3	4									14		
7	8	9	1	2	3	4									14	15		
8	9	1	2	3	4									14	15	16		
9	1	2	3	4				8					14	15	16	17		
10	11	12	13						9	8	7	6						
11	12	13							8	7	6							
12	13								7	6								
13									6									
								14								5		
							14	15							5	4		
						14	15	16						5	4	3		
					14	15	16	17						5	4	3	2	324

csMBC(2,9)

1	2	3	4	5		7	8	9	10	11	12	13	14							158
2	3	4	5		7	8	9	10	1	12	13	14								
3	4	5		7	8	9	10	1	2	13	14									
4	5		7	8	9	10	1	2	3	14										
5		7	8	9	10	1	2	3	4											
	7	8	9	10	1	2	3	4												15
7	8	9	10	1	2	3	4		6									15	16	
8	9	10	1	2	3	4		6										15	16	17
9	10	1	2	3	4		6									15	16	17	18	
10	1	2	3	4		6										15	16	17	18	19
11	12	13	14							10	9	8	7							
12	13	14								9	8	7								
13	14									8	7									
14										7										
									15											6
								15	16									6	5	
							15	16	17								6	5	4	
						15	16	17	18							6	5	4	3	
					15	16	17	18	19						6	5	4	3	2	400

csMBC(2,10)

B.4 General constructions of uniquely completable and critical sets

CDG's Critical Set for  $BC(2,2m+1)$

1	2	...	$m-1$	$m$	$m+1$	$m+2$	$2m-1$	$2m$	$2m+1$	$2m+2$	$2m+3$	...	$3m$	$3m+1$	...	
2	3	...	$m$	$m+1$	$m+2$	$m+2$	$2m$	$2m+1$	1	$2m+3$	$2m+4$		$3m+1$			
3	4	...	$m+1$			...	$2m+1$	1	2	$2m+4$	...		$3m+1$			
⋮										⋮						
$m$	$m+1$	$m+2$	...			$2m+1$	...		$m-1$	$3m+1$						
$m+1$	$m+2$	...	...			1	2		$m$							$3m+2$
$m+2$	$m+3$	...		$2m+1$	$2m+1$	2	...	$m$	$m$							$3m+3$
$m+3$	$m+4$	...	$2m+1$	1	2	...	$m$									⋮
⋮						⋮										
$2m$	$2m+1$	1	2	...		$m$									$3m+2$	$3m+3$
$2m+1$	1	2	...		$m$										4m	$4m+1$
$2m+2$	$2m+3$	...		$3m+1$					1	2			$m-1$	$m$		
$2m+3$	$2m+4$	...	$3m+1$						2	...			$m$			
⋮									⋮							
$3m$	$3m+1$								$m$							
$3m+1$																
									$3m+2$	$3m+3$					$m+1$	$m+2$
									⋮						⋮	
						$3m+2$									$2m-1$	$2m$

Size of set is  $\frac{13n^2-5}{8}$ ,  $n \equiv 2 \pmod{4}$

Critical Set for  $BC(2,2m)$

1	2	...	m-1	m	m+1	m+2	...	2m-1	2m	2m+1	2m+2	...	3m-2	3m-1				
2	3	...	m	m+1	m+2	...	2m-1	2m	1	2m+2	2m+3	...	3m-1					
3	4	...	m+1	m+2	...		2m	1	2	2m+3								
⋮											⋮							
m-1	m		...	2m-2	2m-1	2m	1	...	m-2	3m-1								
m	m+1	...		2m-1	2m	1	2	...	m-1									
m+1	m+2	...		2m	1	2	...		m									3m
m+2	m+3	...	2m	1	2	...		m	m+1						3m	3m+1		
⋮							⋮	⋮	⋮									⋮
2m-1	2m	1	2	...		m		...	2m-2						3m	3m+1	...	4m-2
2m	1	2	...	m-1	m	m+1	...	2m-2	2m-1					3m	3m+1	...	4m-2	4m-1
2m+1	2m+2	...	3m-1							1	2	...	m-2	m-1				
⋮										⋮								
3m-1									m-1									
									3m									m
									3m+1							m	m+1	m+1
									⋮									⋮
									⋮									⋮
									4m-1					m	m+1	...	2m-2	2m-1

Size of set is  $\frac{7n^2}{4}$ ,  $n \equiv 0 \pmod{4}$

Uniquely Completable Set for BCFC(2,2m+1)

1	2	...	m-1	m	m+1	m+2	...	2m-1	2m	2m+1	2m+2	2m+3	...	3m	3m+1	...			
2	3	...	m	m+1	m+2	...	...	2m	2m+1	1	2m+3	2m+4	...	3m+1					
3	4	...	m+1	m+2		...	...	2m	1	2	2m+4	...	...	3m+1					
⋮											⋮								
m	m+1	m+2	...			2m+1	1	2		m-1	3m+1								
m+1	m+2	...	...		2m+1	1	2	...		m									
m+2	m+3	...	2m+1	1	2	...				m								3m+2	3m+3
m+3	m+4	...	2m+1	1	2	...												3m+2	3m+3
⋮							⋮												⋮
2m	2m+1	1	2	...		m										3m+2	3m+3	...	
2m+1	1	2	...		m											3m+3	...	4m	4m+1
2m+2	2m+3	...		3m+1												m+1	m+2	...	2m
2m+3	2m+4	...	3m+1													m+1	m+2	...	2m
⋮																			⋮
3m	3m+1																	m+2	m+3
3m+1																		m+1	m+2
											m+2	m+1	m+2						m+1
										3m+2	m+1	m+2	m+1						
									3m+2	3m+3	m	m+1	m+1						
										⋮	⋮	⋮	⋮						
									4m	4m+1	2	3	...	m	m+1	m+2			

Size of set is  $\frac{13n^2+8n^3}{8}$ ,  $n \equiv 2 \pmod{4}$

Uniquely Completable Set for BCFC(2,2m)

1	2	...	m-1	m	m+1	m+2	...	2m-1	2m	2m+1	2m+2	...	3m-1	3m-2	3m-1				
2	3	...	m	m+1	m+2	...	2m-1	2m	1	2m+2	2m+3	...	3m-1						
3	4	...	m+1	m+2	...		2m	1	2	2m+3	...	3m-1							
⋮											⋮								
m-1	m		...	2m-2	2m-1	2m	1	...	m-2	3m-1									
m	m+1	...		2m-1	2m	1	2	...	m-1										
m+1	m+2	...	...	2m	1	2	...		m									3m	
m+2	m+3	...	2m	1	2	...		m										3m+1	
⋮							⋮											⋮	
2m-1	2m	1	2	...		m							3m+1	...	3m-1	...	4m-2	4m-1	
2m	1	2	...	m-1	m								3m+1	...	3m-1	...	4m-2	4m-1	
2m+1	2m+2	...	3m-1										m+1	m+2	...	2m-1	2m		
⋮																⋮	⋮	⋮	
3m-1																			
										m								m+1	
									3m	m-1	m								
								3m	3m+1	m-2	m-1	m							
									⋮	⋮	⋮								
										2m-1	2m-2	...	m+1	m					

Size of set is  $\frac{13n^2+6n}{8}, n \equiv 0 \pmod{4}$

Uniquely Completable Set  $MBC(2, 2m+1)$  for  $2m+1 \geq 9$

1	2	...	m-1	m	m+1	m+2	...	...	2m-1	2m	2m+1	2m+2	2m+3	...	3m	3m+1	...			
2	3	...	m	m+1	m+2		...	...	2m	2m+1	1	2m+3	2m+4	...	3m+1					
3	4	...	m+1	m+2		...		2m	2m+1	1	2	2m+4	...	3m+1						
⋮												⋮								
m	m+1	m+2	...			2m+1	1	2	...		m-1	3m+1								
m+1	m+2	...	...		2m+1	1	2	...		m										
m+2	m+3	...		2m+1	1	2	...		m										3m+2	3m+3
m+3	m+4	...	2m+1	1	2	...			m										3m+2	3m+3
								⋮			m+4									
⋮								⋮			⋮								⋮	⋮
2m	2m+1	1	2	...		m				2m-1							3m+2	3m+3	...	4m+1
2m+1	1	2	...		m			m+4	...	2m	2m						3m+2	3m+3	...	4m
2m+2	2m+3	...		3m+1								2m+1	2m			m+3	m+2			
2m+3	2m+4	...	3m+1									2m	...	...	m+3	m+2				
⋮												⋮								
3m	3m+1											m+3	m+2							
3m+1												m+2								
											3m+2								m+1	m+1
										3m+2	3m+3								m	m
											⋮								⋮	⋮
											⋮								⋮	⋮
											4m	4m+1						m+1	m	3
								...										...	3	2

Size of the set is  $\frac{14n^2-12n+30}{8}$ ,  $n \equiv 2 \pmod{4}$

Uniquely Completable Set for  $MBC(2,2m)$

1	2	...	$m-1$	$m$	$m+2$	$m+2$	$m+2$	...	$2m-1$	$2m$	$2m+1$	$2m+2$	...	$3m-1$				
2	3	...	$m$	$m+2$	...	$m+2$	$m+2$	$2m-1$	$2m$	1	$2m+2$	$2m+3$	...	$3m-1$				
3	4	...		$m+2$	...			$2m$	1	2	$2m+3$	...						
⋮												⋮						
$m-1$	$m$	...	...	$2m-2$	$2m-1$	$2m$	$2m$	1	...	$m-2$	$3m-1$							
$m$		...		$2m-1$	$2m$	1	2	...	...	$m-1$								
$m+2$	$m+3$	...	$2m$	1	2	...				$m+1$								$3m$
⋮								⋮	⋮									
$2m-1$	$2m$	1	2	...	$m-1$		$m+1$											⋮
$2m$	1	2	...	$m-1$		$m+1$									$3m$	$3m+1$	...	$4m-2$
$2m+1$	$2m+2$	...	$3m-1$								$2m$	$2m-1$	...		$3m$	$3m+1$	$4m-2$	$4m-1$
⋮											⋮	⋮						
$3m-1$											$m+2$							
										$3m$								$m+1$
									$3m$	$3m+1$							$m+1$	$m$
										⋮								⋮
										⋮								⋮
										$4m-2$					$m+1$	$m$	...	3
								...	$4m-2$	$4m-1$								2

Size of the set is  $\frac{13n^2-2n-16}{8}, n \equiv 0 \pmod{4}$



## B.5 Latin Interchanges in $MBC(2,n)$

In this section, latin interchanges for modified-two backcirculant  $MBC(2,n)$  latin squares,  $n \leq 25$ , are given which have been found using computer search:

Latin Interchanges for MBC(2,3)  
.....

MBC(2,3)

1	2	3	4	5	6
2	3	1	5	6	4
3	1	2	6	4	5
4	5	6	3	2	1
5	6	4	2	1	3
6	4	5	1	3	2

Critical Set

1	2	3	4	*	*
2	3	1	*	*	*
3	1	*	*	*	5
4	*	*	3	*	*
*	*	*	*	*	*
*	*	5	*	*	2

LI(1,1)

5*	2	3	4	1*	6
2	3	1	5	6	4
3	1	2	6	4	5
4	5	6	3	2	1
1*	6	4	2	5*	3
6	4	5	1	3	2

LI(1,2)

1	5*	3	4	2*	6
2	3	1	5	6	4
3	1	2	6	4	5
4	2*	6	3	5*	1
5	6	4	2	1	3
6	4	5	1	3	2

LI(1,3)

1	2	6*	4	5	3*
2	3	1	5	4*	6*
3	1	4*	6	2*	5
4	5	2*	3	6*	1
5	6	3*	2	1	4*
6	4	5	1	3	2

LI(2,1)

1	2	3	4	5	6
5*	3	1	2*	6	4
3	1	2	6	4	5
4	5	6	3	2	1
2*	6	4	5*	1	3
6	4	5	1	3	2

LI(2,2)

1	2	3	4	5	6
---	---	---	---	---	---

2	6*	1	5	4*	3*
3	1	4*	6	2*	5
4	5	2*	3	6*	1
5	3*	6*	2	1	4*
6	4	5	1	3	2

LI(2,3)

1	2	3	4	5	6
2	3	6*	5	1*	4
3	1	4*	6	2*	5
4	5	2*	3	6*	1
5	6	1*	2	4*	3
6	4	5	1	3	2

Latin interchanges for MBC(2,5)

.....

MBC(2,5)

1	2	3	4	5	6	7	8	9	10
2	3	4	5	1	7	8	9	10	6
3	4	5	1	2	8	9	10	6	7
4	5	1	2	3	9	10	6	7	8
5	1	2	3	4	10	6	7	8	9
6	7	8	9	10	5	4	3	2	1
7	8	9	10	6	4	3	2	1	5
8	9	10	6	7	3	2	1	5	4
9	10	6	7	8	2	1	5	4	3
10	6	7	8	9	1	5	4	3	2

Critical Set

1	2	3	4	5	6	7	*	*	*
2	3	4	5	1	7	*	*	*	*
3	4	5	1	2	*	*	*	*	*
4	5	1	2	*	*	*	*	*	8
5	1	2	*	*	*	*	*	8	9
6	7	*	*	*	5	4	*	*	*
7	*	*	*	*	4	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	8	*	*	*	*	3
*	*	*	8	9	*	*	*	3	2

LI(1,1) in S1

8*	2	3	4	5	6	7	1*	9	10
2	3	4	5	1	7	8	9	10	6
3	4	5	1	2	8	9	10	6	7
4	5	1	2	3	9	10	6	7	8
5	1	2	3	4	10	6	7	8	9
6	7	8	9	10	5	4	3	2	1
7	8	9	10	6	4	3	2	1	5
1*	9	10	6	7	3	2	8*	5	4
9	10	6	7	8	2	1	5	4	3
10	6	7	8	9	1	5	4	3	2

LI(1,2) in S1

1	8*	3	4	5	6	7	2*	9	10
2	3	4	5	1	7	8	9	10	6
3	4	5	1	2	8	9	10	6	7
4	5	1	2	3	9	10	6	7	8
5	1	2	3	4	10	6	7	8	9
6	7	8	9	10	5	4	3	2	1
7	2*	9	10	6	4	3	8*	1	5
8	9	10	6	7	3	2	1	5	4
9	10	6	7	8	2	1	5	4	3
10	6	7	8	9	1	5	4	3	2

LI(1,3) in S1

1	2	8*	4	5	6	7	3*	9	10
2	3	4	5	1	7	8	9	10	6
3	4	5	1	2	8	9	10	6	7
4	5	1	2	3	9	10	6	7	8
5	1	2	3	4	10	6	7	8	9
6	7	3*	9	10	5	4	8*	2	1
7	8	9	10	6	4	3	2	1	5
8	9	10	6	7	3	2	1	5	4
9	10	6	7	8	2	1	5	4	3
10	6	7	8	9	1	5	4	3	2

LI(1,4) in S3

1	2	3	10*	5	6	7	8	9	4*
2	3	4	5	1	7	8	9	10	6
3	4	5	1	2	8	9	10	6	7
4	5	1	2	3	9	10	6	7	8
5	1	2	6*	4	10	3*	7	8	9
6	7	8	9	10	5	4	3	2	1
7	8	9	3*	6	4	2*	1*	5*	10*
8	9	10	4*	7	3	6*	2*	1*	5*
9	10	6	7	8	2	1	5	4	3
10	6	7	8	9	1	5	4	3	2

LI(1,5) in S3

1	2	3	4	10*	6	7	8	9	5*
2	3	4	5	1	7	8	9	10	6
3	4	5	1	2	8	9	10	6	7
4	5	1	2	6*	9	10	3*	7	8
5	1	2	3	4	10	6	7	8	9
6	7	8	9	3*	5	4	2*	1*	10*
7	8	9	10	5*	4	3	6*	2*	1*
8	9	10	6	7	3	2	1	5	4
9	10	6	7	8	2	1	5	4	3
10	6	7	8	9	1	5	4	3	2

LI(2,2) in S1

1	2	3	4	5	6	7	8	9	10
2	8*	4	5	1	7	3*	9	10	6
3	4	5	1	2	8	9	10	6	7
4	5	1	2	3	9	10	6	7	8
5	1	2	3	4	10	6	7	8	9
6	7	8	9	10	5	4	3	2	1
7	3*	9	10	6	4	8*	2	1	5
8	9	10	6	7	3	2	1	5	4
9	10	6	7	8	2	1	5	4	3
10	6	7	8	9	1	5	4	3	2

LI(2,3) in S2 having pattern P0

1	2	3	4	5	6	7	8	9	10
2	3	10*	5	1	7	8	9	4*	6
3	4	5	1	2	8	9	10	6	7
4	5	1	2	3	9	10	6	7	8
5	1	2	6*	4	10	3*	7	8	9

Latin Interchanges for MBC(2,7)  
 .....

MBC(2,7)

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	4	5	6	14	8	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	14	8	6	5	4	3	2	1	7
10	11	12	13	14	8	9	5	4	3	2	1	7	6
11	12	13	14	8	9	10	4	3	2	1	7	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

Critical Set

1	2	3	4	5	6	7	8	9	10	**	**	**	**
2	3	4	5	6	7	1	9	10	**	**	**	**	**
3	4	5	6	7	1	2	10	**	**	**	**	**	**
4	5	6	7	1	2	3	**	**	**	**	**	**	**
5	6	7	1	2	3	**	**	**	**	**	**	**	11
6	7	1	2	3	**	**	**	**	**	**	**	11	12
7	1	2	3	**	**	**	**	**	**	**	11	12	13
8	9	10	**	**	**	**	7	6	5	**	**	**	**
9	10	**	**	**	**	**	6	5	**	**	**	**	**
10	**	**	**	**	**	**	5	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	**	**	11	**	**	**	**	**	**	4
**	**	**	**	**	11	12	**	**	**	**	**	4	3
**	**	**	**	11	12	13	**	**	**	**	4	3	2

LI(1,5) in S3

1	2	3	4	14*	6	7	8	9	10	11	12	13	5*
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	8*	5	6	14	4*	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	14	8	6	5	4	3	2	1	7
10	11	12	13	4*	8	9	5	3*	2*	1*	7*	6*	14*
11	12	13	14	5*	9	10	4	8*	3*	2*	1*	7*	6*
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(2,4) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	14*	6	7	1	9	10	11	12	13	5*	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9

4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	8*	5	6	14	4*	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	14	8	6	5	4	3	2	1	7
10	11	12	13	4*	8	9	5	2*	3	7*	1	14*	6
11	12	13	8*	14*	9	10	4	3	2	1	7	6	5
12	13	14	5*	9	10	11	3	8*	1	2*	6	7*	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(3,3) in S2 having pattern P0

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	14*	6	7	1	2	10	11	12	13	5*	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	8*	5	6	14	4*	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	14	8	6	5	4	3	2	1	7
10	11	12	13	4*	8	9	5	1*	3	2	14*	7	6
11	12	13	8*	14*	9	10	4	3	2	1	7	6	5
12	13	8*	14*	9	10	11	3	2	1	7	6	5	4
13	14	5*	9	10	11	12	2	8*	7	6	1*	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(1,6) in S3

1	2	3	4	5	14*	7	8	9	10	11	12	13	6*
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	8*	5	13	14	4*	9	10	11	12
7	1	2	3	4	5	6	14	8	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	4*	8	6	5	3*	2*	1*	7*	14*
10	11	12	13	14	6*	9	5	4	8*	3*	2*	1*	7*
11	12	13	14	8	9	10	4	3	2	1	7	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(2,5) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	14*	7	1	9	10	11	12	13	6*	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	8*	5	6	14	4*	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	14	8	6	5	4	3	2	1	7
10	11	12	13	4*	8	9	5	3*	2*	1*	7*	14*	6
11	12	13	14	6*	9	10	4	8*	3*	2*	1*	7*	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4

13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(3,4) in S2 having pattern P3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	14*	7	1	2	10	11	12	13	6*	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	8*	5	6	14	4*	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	14	8	6	5	4	3	2	1	7
10	11	12	13	4*	8	9	5	2*	3	1*	14*	7	6
11	12	13	8*	14*	9	10	4	3	1*	2*	7	6	5
12	13	14	6*	9	10	11	3	8*	2*	7	1*	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(1,7) in S3

1	2	3	4	5	6	14*	8	9	10	11	12	13	7*
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	8*	12	13	14	4*	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	4	5	6	14	8	9	10	11	12	13
8	9	10	11	12	13	4*	7	6	5	3*	2*	1*	14*
9	10	11	12	13	14	7*	6	5	4	8*	3*	2*	1*
10	11	12	13	14	8	9	5	4	3	2	1	7	6
11	12	13	14	8	9	10	4	3	2	1	7	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(2,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	14*	1	9	10	11	12	13	7*	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	8*	5	13	14	4*	9	10	11	12
7	1	2	3	4	5	6	14	8	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	4*	8	6	5	3*	2*	1*	14*	7
10	11	12	13	14	7*	9	5	4	8*	3*	2*	1*	6
11	12	13	14	8	9	10	4	3	2	1	7	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(3,5) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	14*	1	2	10	11	12	13	7*	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10



5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	8*	5	6	14	4*	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	14	8	6	5	4	3	2	1	7
10	11	12	13	4*	8	9	5	3*	2*	1*	14*	7	6
11	12	13	14	7*	9	10	4	8*	3*	2*	1*	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(4,4) in S2 having pattern P0

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	14*	1	2	3	11	12	13	7*	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	8*	5	6	14	4*	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	14	8	6	5	4	3	2	1	7
10	11	12	13	4*	8	9	5	2*	3	14*	1	7	6
11	12	13	8*	14*	9	10	4	3	2	1	7	6	5
12	13	14	7*	9	10	11	3	8*	1	2*	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(2,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	14*	9	10	11	12	13	1*	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	8*	12	13	14	4*	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	4	5	6	14	8	9	10	11	12	13
8	9	10	11	12	13	4*	7	6	5	3*	2*	14*	1
9	10	11	12	13	14	1*	6	5	4	8*	3*	2*	7
10	11	12	13	14	8	9	5	4	3	2	1	7	6
11	12	13	14	8	9	10	4	3	2	1	7	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(3,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	7	14*	2	10	11	12	13	1*	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	8*	5	13	14	4*	9	10	11	12
7	1	2	3	4	5	6	14	8	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	4*	8	6	5	3*	2*	14*	1	7
10	11	12	13	14	1*	9	5	4	8*	3*	2*	7	6
11	12	13	14	8	9	10	4	3	2	1	7	6	5

12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(4,5) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	14*	2	3	11	12	13	1*	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	8*	5	6	14	4*	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	14	8	6	5	4	3	2	1	7
10	11	12	13	4*	8	9	5	3*	2*	14*	1	7	6
11	12	13	14	1*	9	10	4	8*	3*	2*	7	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(3,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	7	1	14*	10	11	12	13	2*	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	3	8*	12	13	14	4*	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	4	5	6	14	8	9	10	11	12	13
8	9	10	11	12	13	4*	7	6	5	3*	14*	2	1
9	10	11	12	13	14	2*	6	5	4	8*	3*	1	7
10	11	12	13	14	8	9	5	4	3	2	1	7	6
11	12	13	14	8	9	10	4	3	2	1	7	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(4,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	1	14*	3	11	12	13	2*	8	9	10
5	6	7	1	2	3	4	12	13	14	8	9	10	11
6	7	1	2	3	8*	5	13	14	4*	9	10	11	12
7	1	2	3	4	5	6	14	8	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	4*	8	6	5	3*	14*	2	1	7
10	11	12	13	14	2*	9	5	4	8*	3*	1	7	6
11	12	13	14	8	9	10	4	3	2	1	7	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(5,5) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8

3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	14*	3	4	12	13	2*	8	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	8*	5	6	14	4*	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	14	8	6	5	4	3	2	1	7
10	11	12	13	4*	8	9	5	3*	14*	2	1	7	6
11	12	13	14	2*	9	10	4	8*	3*	1	7	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(4,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	1	2	14*	11	12	13	3*	8	9	10
5	6	7	1	2	3	8*	12	13	14	4*	9	10	11
6	7	1	2	3	4	5	13	14	8	9	10	11	12
7	1	2	3	4	5	6	14	8	9	10	11	12	13
8	9	10	11	12	13	4*	7	6	5	14*	3	2	1
9	10	11	12	13	14	3*	6	5	4	8*	2	1	7
10	11	12	13	14	8	9	5	4	3	2	1	7	6
11	12	13	14	8	9	10	4	3	2	1	7	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

LI(5,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	4	5	6	7	1	9	10	11	12	13	14	8
3	4	5	6	7	1	2	10	11	12	13	14	8	9
4	5	6	7	1	2	3	11	12	13	14	8	9	10
5	6	7	1	2	14*	4	12	13	3*	8	9	10	11
6	7	1	2	3	8*	5	13	14	4*	9	10	11	12
7	1	2	3	4	5	6	14	8	9	10	11	12	13
8	9	10	11	12	13	14	7	6	5	4	3	2	1
9	10	11	12	13	4*	8	6	5	14*	3	2	1	7
10	11	12	13	14	3*	9	5	4	8*	2	1	7	6
11	12	13	14	8	9	10	4	3	2	1	7	6	5
12	13	14	8	9	10	11	3	2	1	7	6	5	4
13	14	8	9	10	11	12	2	1	7	6	5	4	3
14	8	9	10	11	12	13	1	7	6	5	4	3	2

Latin Interchanges for mbc(2,9)  
.....

MBC(2,9)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

Critical Set

1	2	3	4	5	6	7	8	9	10	11	12	13	**	**	**	**	**
2	3	4	5	6	7	8	9	1	11	12	13	**	**	**	**	**	**
3	4	5	6	7	8	9	1	2	12	13	**	**	**	**	**	**	**
4	5	6	7	8	9	1	2	3	13	**	**	**	**	**	**	**	**
5	6	7	8	9	1	2	3	4	**	**	**	**	**	**	**	**	**
6	7	8	9	1	2	3	4	**	**	**	**	**	**	**	**	**	14
7	8	9	1	2	3	4	**	**	**	**	**	**	**	**	**	**	14 15
8	9	1	2	3	4	**	**	**	**	**	**	**	**	**	**	**	14 15 16
9	1	2	3	4	**	**	**	8	**	**	**	**	**	**	**	**	14 15 16 17
10	11	12	13	**	**	**	**	**	9	8	7	6	**	**	**	**	**
11	12	13	**	**	**	**	**	**	8	7	6	**	**	**	**	**	**
12	13	**	**	**	**	**	**	**	7	6	**	**	**	**	**	**	**
13	**	**	**	**	**	**	**	**	6	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	14	**	**	**	**	**	**	7	** 5
**	**	**	**	**	**	**	**	14	15	**	**	**	**	**	**	**	5 4
**	**	**	**	**	**	**	14	15	16	**	**	**	**	**	**	5	4 3
**	**	**	**	**	14	15	16	17	**	**	**	**	**	5	4	3	2

LI(1,6)in S3

1	2	3	4	5	18*	7	8	9	10	11	12	13	14	15	16	17	6*
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*	10	11	12	6	4*	3*	2*	1*	9*	8*	7*	18*

14	15	16	17	18	6*11	12	13	5	10*	4*	3*	2*	1*	9*	8*	7*	
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(2,5) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	18*	7	8	9	1	11	12	13	14	15	16	17	6*10	
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	3*	4	1*	2	8*	9	18*	7	
14	15	16	17	10*18*	11	12	13	5	4	3	2	1	9	8	7	6	
15	16	17	18	6*11	12	13	14	4	10*	2	3*	9	1*	7	8*	5	
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,4) in S2 having pattern P4

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	18*	7	8	9	1	2	12	13	14	15	16	17	6*10	11	
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	2*	4	3	9*	1	18*	8	7	
14	15	16	17	10*18*	11	12	13	5	4	3	1*	2*	9	8	7	6	
15	16	17	10*18*	11	12	13	14	4	3	2	9*	1*	8	7	6	5	
16	17	18	6*11	12	13	14	15	3	10*	1	2*	8	7	9*	5	4	
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,3) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	18*	7	8	9	1	2	3	13	14	15	16	17	6*10	11	12	
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1

11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	1*	4	3	2	18*	9	8	7	
14	15	16	17	10*18*	11	12	13	5	4	3	2	1	9	8	7	6	
15	16	17	10*18*	11	12	13	14	4	3	2	1	9	8	7	6	5	
16	17	10*18*	11	12	13	14	15	3	2	1	9	8	7	6	5	4	
17	18	6*11	12	13	14	15	16	2	10*	9	8	7	1*	5	4	3	
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(1,7) in S3

1	2	3	4	5	6	18*	8	9	10	11	12	13	14	15	16	17	7*
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*11	12	13	14	15	16	
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*10	11	7	6	4*	3*	2*	1*	9*	8*18*		
13	14	15	16	17	18	7*11	12	6	5	10*	4*	3*	2*	1*	9*	8*	8*
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(2,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	18*	8	9	1	11	12	13	14	15	16	17	7*10	
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	4*	3*	2*	1*	9*	8*18*	7		
14	15	16	17	18	7*11	12	13	5	10*	4*	3*	2*	1*	9*	8*	6	
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,5) in S2 having pattern P2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	18*	8	9	1	2	12	13	14	15	16	17	7*10	11	
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15

8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*	10	11	12	6	3*	4	2*	9*	1	18*	8	7
14	15	16	17	10*	18*	11	12	13	5	4	3	1*	2*	9	8	7	6
15	16	17	18	7*	11	12	13	14	4	10*	2	3*	1*	8	9*	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,4) in S2 having pattern P3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	18*	8	9	1	2	3	13	14	15	16	17	7*	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*	10	11	12	6	2*	4	3	1*	18*	9	8	7
14	15	16	17	10*	18*	11	12	13	5	4	3	1*	2*	9	8	7	6
15	16	17	10	18*	11	12	13	14	4	3	1*	2*	9	8	7	6	5
16	17	18	7*	11	12	13	14	15	3	10*	2*	9	8	1*	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(1,8) in S3

1	2	3	4	5	6	7	18*	9	10	11	12	13	14	15	16	17	8*
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	10*	6	16	17	18	5*	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	5*	10	8	7	6	4*	3*	2*	1*	9*	18*
12	13	14	15	16	17	18	8*	11	7	6	5	10*	4*	3*	2*	1*	9*
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(2,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	18*	9	1	11	12	13	14	15	16	17	8*	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12

5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*11	12	13	14	15	16	
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*10	11	7	6	4*	3*	2*	1*	9*18*	8		
13	14	15	16	17	18	8*11	12	6	5	10*	4*	3*	2*	1*	9*	7	
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	18*	9	1	2	12	13	14	15	16	17	8*10	11	
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	4*	3*	2*	1*	9*18*	8	7		
14	15	16	17	18	8*11	12	13	5	10*	4*	3*	2*	1*	9*	7	6	
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,5) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	18*	9	1	2	3	13	14	15	16	17	8*10	11	12	
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	3*	4	1*	2	18*	9	8	7	
14	15	16	17	10*18*	11	12	13	5	4	3	2	1	9	8	7	6	
15	16	17	18	8*11	12	13	14	4	10*	2	3*	9	1*	7	6	5	
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(1,9) in S3

1	2	3	4	5	6	7	8	18*10	11	12	13	14	15	16	17	9*
---	---	---	---	---	---	---	---	-------	----	----	----	----	----	----	----	----



2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	10*15	16	17	18	5*11	12	13	14		
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	5*	9	8	7	6	4*	3*	2*	1*18*	
11	12	13	14	15	16	17	18	9*	8	7	6	5	10*	4*	3*	2*	1*
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(2,8) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	18*	1	11	12	13	14	15	16	17	9*10	
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	10*	6	16	17	18	5*11	12	13	14	15	
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	5*10	8	7	6	4*	3*	2*	1*18*	9		
12	13	14	15	16	17	18	9*11	7	6	5	10*	4*	3*	2*	1*	8	
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	18*	1	2	12	13	14	15	16	17	9*10	11	
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*11	12	13	14	15	16	
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*10	11	7	6	4*	3*	2*	1*18*	9	8		
13	14	15	16	17	18	9*11	12	6	5	10*	4*	3*	2*	1*	8	7	
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	18*	1	2	3	13	14	15	16	17	9*	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*	10	11	12	6	4*	3*	2*	1*	18*	9	8	7
14	15	16	17	18	9*	11	12	13	5	10*	4*	3*	2*	1*	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(5,5) in S2 having pattern P3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	18*	1	2	3	4	14	15	16	17	9*	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*	10	11	12	6	3*	4	2*	18*	1	9	8	7
14	15	16	17	10*	18*	11	12	13	5	4	2*	3*	1	9	8	7	6
15	16	17	18	9*	11	12	13	14	4	10*	3*	1	2*	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(2,9) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	18*	11	12	13	14	15	16	17	1*	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	10*	15	16	17	18	5*	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	5*	9	8	7	6	4*	3*	2*	18*	1
11	12	13	14	15	16	17	18	1*	8	7	6	5	10*	4*	3*	2*	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5

16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,8) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	18*	2	12	13	14	15	16	17	1*	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	10*	6	16	17	18	5*	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	5*	10	8	7	6	4*	3*	2*	18*	1	9
12	13	14	15	16	17	18	1*	11	7	6	5	10*	4*	3*	2*	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	18*	2	3	13	14	15	16	17	1*	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*	10	11	7	6	4*	3*	2*	18*	1	9	8
13	14	15	16	17	18	1*	11	12	6	5	10*	4*	3*	2*	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(5,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	18*	2	3	4	14	15	16	17	1*	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8

13	14	15	16	17	5*10	11	12	6	4*	3*	2*18*	1	9	8	7
14	15	16	17	18	1*11	12	13	5	10*	4*	3*	2*	9	8	7
15	16	17	18	10	11	12	13	4	3	2	1	9	8	7	6
16	17	18	10	11	12	13	14	3	2	1	9	8	7	6	5
17	18	10	11	12	13	14	15	2	1	9	8	7	6	5	4
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4
															3
															2

LI(6,5) in S2 having pattern P0

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	18*	2	3	4	5	15	16	17	1*	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	3*	4	18*	2	1	9	8	7	
14	15	16	17	10*18*	11	12	13	5	4	3	2	1	9	8	7	6	
15	16	17	18	1*11	12	13	14	4	10*	2	3*	9	8	7	6	5	
16	17	18	10	11	12	13	14	3	2	1	9	8	7	6	5	4	
17	18	10	11	12	13	14	15	2	1	9	8	7	6	5	4	3	
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,9) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	18*	12	13	14	15	16	17	2*	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	10*	15	16	17	18	5*	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	5*	9	8	7	6	4*	3*18*	2	1	
11	12	13	14	15	16	17	18	2*	8	7	6	5	10*	4*	3*	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,8) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	18*	3	13	14	15	16	17	2*	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	10*	6	16	17	18	5*	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16

9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	5*10		8	7	6	4*	3*18*	2	1	9	
12	13	14	15	16	17	18	2*11		7	6	5	10*	4*	3*	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(5,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	18*	3	4	14	15	16	17	2*10	11	12	13	
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*11	12	13	14	15	16	
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*10	11		7	6	4*	3*18*	2	1	9	8	
13	14	15	16	17	18	2*11	12		6	5	10*	4*	3*	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(6,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	18*	3	4	5	15	16	17	2*10	11	12	13	14	
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12		6	4*	3*18*	2	1	9	8	7	
14	15	16	17	18	2*11	12	13		5	10*	4*	3*	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,9) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	18*13	14	15	16	17	3*10	11	12		
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13

6	7	8	9	1	2	3	4	10*	15	16	17	18	5*	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	5*	9	8	7	6	4*	18*	3	2	1
11	12	13	14	15	16	17	18	3*	8	7	6	5	10*	4*	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(5,8) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	18*	4	14	15	16	17	3*	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	10*	6	16	17	18	5*	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	5*	10	8	7	6	4*	18*	3	2	1	9
12	13	14	15	16	17	18	3*	11	7	6	5	10*	4*	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(6,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	18*	4	5	15	16	17	3*	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*	10	11	7	6	4*	18*	3	2	1	9	8
13	14	15	16	17	18	3*	11	12	6	5	10*	4*	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(5,9) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	18*	14	15	16	17	4*	10	11	12	13
6	7	8	9	1	2	3	4	10*	15	16	17	18	5*	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	5*	9	8	7	6	18*	4	3	2	1
11	12	13	14	15	16	17	18	4*	8	7	6	5	10*	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(7,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	18*	5	6	16	17	4*	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*	10	11	7	6	18*	4	3	2	1	9	8
13	14	15	16	17	18	4*	11	12	6	5	10*	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(9,9) in S4 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	13*	16	17	18	10	11	12	6*	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	13*	7	6*	18	10	11	12	8*	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	8*	5	4	3	2	1	9	13*	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5

16	17	18	10	11	12	6*	14	15	3	2	1	9	13*	7	8*	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2



Latin Interchanges for mbc(2,9)  
.....

MBC(2,9)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

Critical Set

1	2	3	4	5	6	7	8	9	10	11	12	13	**	**	**	**	**
2	3	4	5	6	7	8	9	1	11	12	13	**	**	**	**	**	**
3	4	5	6	7	8	9	1	2	12	13	**	**	**	**	**	**	**
4	5	6	7	8	9	1	2	3	13	**	**	**	**	**	**	**	**
5	6	7	8	9	1	2	3	4	**	**	**	**	**	**	**	**	**
6	7	8	9	1	2	3	4	**	**	**	**	**	**	**	**	**	14
7	8	9	1	2	3	4	**	**	**	**	**	**	**	**	**	**	14 15
8	9	1	2	3	4	**	**	**	**	**	**	**	**	**	**	**	14 15 16
9	1	2	3	4	**	**	**	8	**	**	**	**	**	**	**	**	14 15 16 17
10	11	12	13	**	**	**	**	**	9	8	7	6	**	**	**	**	**
11	12	13	**	**	**	**	**	**	8	7	6	**	**	**	**	**	**
12	13	**	**	**	**	**	**	**	7	6	**	**	**	**	**	**	**
13	**	**	**	**	**	**	**	**	6	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	14	**	**	**	**	**	**	7	** 5
**	**	**	**	**	**	**	**	14	15	**	**	**	**	**	**	**	5 4
**	**	**	**	**	**	14	15	16	**	**	**	**	**	**	**	5	4 3
**	**	**	**	**	14	15	16	17	**	**	**	**	**	**	5	4	3 2

LI(1,6)in S3

1	2	3	4	5	18*	7	8	9	10	11	12	13	14	15	16	17	6*
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*	10	11	12	6	4*	3*	2*	1*	9*	8*	7*	18*

14	15	16	17	18	6*11	12	13	5	10*	4*	3*	2*	1*	9*	8*	7*
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3

LI(2,5) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	18*	7	8	9	1	11	12	13	14	15	16	17	6*10	
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	3*	4	1*	2	8*	9	18*	7	
14	15	16	17	10*18*11	12	13	5	4	3	2	1	9	8	7	6		
15	16	17	18	6*11	12	13	14	4	10*	2	3*	9	1*	7	8*	5	
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,4) in S2 having pattern P4

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	18*	7	8	9	1	2	12	13	14	15	16	17	6*10	11	
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	2*	4	3	9*	1	18*	8	7	
14	15	16	17	10*18*11	12	13	5	4	3	1*	2*	9	8	7	6		
15	16	17	10*18*11	12	13	14	4	3	2	9*	1*	8	7	6	5		
16	17	18	6*11	12	13	14	15	3	10*	1	2*	8	7	9*	5	4	
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,3) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	18*	7	8	9	1	2	3	13	14	15	16	17	6*10	11	12	
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1

11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	1*	4	3	2	18*	9	8	7	
14	15	16	17	10*18*	11	12	13	5	4	3	2	1	9	8	7	6	
15	16	17	10*18*	11	12	13	14	4	3	2	1	9	8	7	6	5	
16	17	10*18*	11	12	13	14	15	3	2	1	9	8	7	6	5	4	
17	18	6*11	12	13	14	15	16	2	10*	9	8	7	1*	5	4	3	
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(1,7) in S3

1	2	3	4	5	6	18*	8	9	10	11	12	13	14	15	16	17	7*
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*11	12	13	14	15	16	
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*10	11	7	6	4*	3*	2*	1*	9*	8*18*		
13	14	15	16	17	18	7*11	12	6	5	10*	4*	3*	2*	1*	9*	8*	
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(2,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	18*	8	9	1	11	12	13	14	15	16	17	7*10	
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	4*	3*	2*	1*	9*	8*18*	7		
14	15	16	17	18	7*11	12	13	5	10*	4*	3*	2*	1*	9*	8*	6	
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,5) in S2 having pattern P2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	18*	8	9	1	2	12	13	14	15	16	17	7*10	11	
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15

8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*	10	11	12	6	3*	4	2*	9*	1	18*	8	7
14	15	16	17	10*	18*	11	12	13	5	4	3	1*	2*	9	8	7	6
15	16	17	18	7*	11	12	13	14	4	10*	2	3*	1*	8	9*	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,4) in S2 having pattern P3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	18*	8	9	1	2	3	13	14	15	16	17	7*	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*	10	11	12	6	2*	4	3	1*	18*	9	8	7
14	15	16	17	10*	18*	11	12	13	5	4	3	1*	2*	9	8	7	6
15	16	17	10	18*	11	12	13	14	4	3	1*	2*	9	8	7	6	5
16	17	18	7*	11	12	13	14	15	3	10*	2*	9	8	1*	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(1,8) in S3

1	2	3	4	5	6	7	18*	9	10	11	12	13	14	15	16	17	8*
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	10*	6	16	17	18	5*	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	5*	10	8	7	6	4*	3*	2*	1*	9*	18*
12	13	14	15	16	17	18	8*	11	7	6	5	10*	4*	3*	2*	1*	9*
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(2,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	18*	9	1	11	12	13	14	15	16	17	8*	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12

5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*11	12	13	14	15	16	
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*10	11	7	6	4*	3*	2*	1*	9*18*	8		
13	14	15	16	17	18	8*11	12	6	5	10*	4*	3*	2*	1*	9*	7	
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	18*	9	1	2	12	13	14	15	16	17	8*10	11	
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	4*	3*	2*	1*	9*18*	8	7		
14	15	16	17	18	8*11	12	13	5	10*	4*	3*	2*	1*	9*	7	6	
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,5) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	18*	9	1	2	3	13	14	15	16	17	8*10	11	12	
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	3*	4	1*	2	18*	9	8	7	
14	15	16	17	10*18*11	12	13	5	4	3	2	1	9	8	7	6		
15	16	17	18	8*11	12	13	4	10*	2	3*	9	1*	7	6	5		
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(1,9) in S3

1	2	3	4	5	6	7	8	18*10	11	12	13	14	15	16	17	9*
---	---	---	---	---	---	---	---	-------	----	----	----	----	----	----	----	----

2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	10*15	16	17	18	5*11	12	13	14		
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	5*	9	8	7	6	4*	3*	2*	1*18*	
11	12	13	14	15	16	17	18	9*	8	7	6	5	10*	4*	3*	2*	1*
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(2,8) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	18*	1	11	12	13	14	15	16	17	9*10	
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	10*	6	16	17	18	5*11	12	13	14	15	
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	5*10	8	7	6	4*	3*	2*	1*18*	9		
12	13	14	15	16	17	18	9*11	7	6	5	10*	4*	3*	2*	1*	8	
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	18*	1	2	12	13	14	15	16	17	9*10	11	
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*11	12	13	14	15	16	
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*10	11	7	6	4*	3*	2*	1*18*	9	8		
13	14	15	16	17	18	9*11	12	6	5	10*	4*	3*	2*	1*	8	7	
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	18*	1	2	3	13	14	15	16	17	9*10	11	12	
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	4*	3*	2*	1*18*	9	8	7		
14	15	16	17	18	9*11	12	13	5	10*	4*	3*	2*	1*	8	7	6	
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(5,5) in S2 having pattern P3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	18*	1	2	3	4	14	15	16	17	9*10	11	12	13	
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	3*	4	2*18*	1	9	8	7		
14	15	16	17	10*18*11	12	13	5	4	2*	3*	1	9	8	7	6	5	
15	16	17	18	9*11	12	13	14	4	10*	3*	1	2*	8	7	6	5	
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(2,9) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	18*11	12	13	14	15	16	17	1*	10	
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	10*15	16	17	18	5*11	12	13	14		
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	5*	9	8	7	6	4*	3*	2*18*	1	
11	12	13	14	15	16	17	18	1*	8	7	6	5	10*	4*	3*	2*	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5

16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,8) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	18*	2	12	13	14	15	16	17	1*	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	10*	6	16	17	18	5*	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	5*	10	8	7	6	4*	3*	2*	18*	1	9
12	13	14	15	16	17	18	1*	11	7	6	5	10*	4*	3*	2*	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	18*	2	3	13	14	15	16	17	1*	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*	10	11	7	6	4*	3*	2*	18*	1	9	8
13	14	15	16	17	18	1*	11	12	6	5	10*	4*	3*	2*	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(5,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	18*	2	3	4	14	15	16	17	1*	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8



13	14	15	16	17	5*10	11	12	6	4*	3*	2*18*	1	9	8	7
14	15	16	17	18	1*11	12	13	5	10*	4*	3*	2*	9	8	7
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4

LI(6,5) in S2 having pattern P0

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	18*	2	3	4	5	15	16	17	1*10	11	12	13	14	15
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	18
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	3*	4	18*	2	1	9	8	7	6
14	15	16	17	10*18*11	12	13	5	4	3	2	1	9	8	7	6	5	4
15	16	17	18	1*11	12	13	14	4	10*	2	3*	9	8	7	6	5	4
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(3,9) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	18*12	13	14	15	16	17	2*10	11	12	13
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	10*15	16	17	18	5*11	12	13	14	15	16
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	5*	9	8	7	6	4*	3*18*	2	1	9
11	12	13	14	15	16	17	18	2*	8	7	6	5	10*	4*	3*	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,8) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	18*	3	13	14	15	16	17	2*10	11	12	13
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	10*	6	16	17	18	5*11	12	13	14	15	16
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16

9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	5*10	8	7	6	4*	3*18*	2	1	9		
12	13	14	15	16	17	18	2*11	7	6	5	10*	4*	3*	1	9	8	
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(5,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	18*	3	4	14	15	16	17	2*10	11	12	13	
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*11	12	13	14	15	16	
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*10	11	7	6	4*	3*18*	2	1	9	8		
13	14	15	16	17	18	2*11	12	6	5	10*	4*	3*	1	9	8	7	
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(6,6) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	18*	3	4	5	15	16	17	2*10	11	12	13	14	
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	10*	6	7	8	18	5*11	12	13	14	15	16	17	
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	5*10	11	12	6	4*	3*18*	2	1	9	8	7		
14	15	16	17	18	2*11	12	13	5	10*	4*	3*	1	9	8	7	6	
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(4,9) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	18*13	14	15	16	17	3*10	11	12		
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13

6	7	8	9	1	2	3	4	10*	15	16	17	18	5*	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	5*	9	8	7	6	4*	18*	3	2	1
11	12	13	14	15	16	17	18	3*	8	7	6	5	10*	4*	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(5,8) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	18*	4	14	15	16	17	3*	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	10*	6	16	17	18	5*	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	5*	10	8	7	6	4*	18*	3	2	1	9
12	13	14	15	16	17	18	3*	11	7	6	5	10*	4*	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(6,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	18*	4	5	15	16	17	3*	10	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*	10	11	7	6	4*	18*	3	2	1	9	8
13	14	15	16	17	18	3*	11	12	6	5	10*	4*	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(5,9) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	18*	14	15	16	17	4*	10	11	12	13
6	7	8	9	1	2	3	4	10*	15	16	17	18	5*	11	12	13	14
7	8	9	1	2	3	4	5	6	16	17	18	10	11	12	13	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	5*	9	8	7	6	18*	4	3	2	1
11	12	13	14	15	16	17	18	4*	8	7	6	5	10*	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(7,7) in S3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	18*	5	6	16	17	4*	10	11	12	13	14	15
8	9	1	2	3	4	10*	6	7	17	18	5*	11	12	13	14	15	16
9	1	2	3	4	5	6	7	8	18	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	5*	10	11	7	6	18*	4	3	2	1	9	8
13	14	15	16	17	18	4*	11	12	6	5	10*	3	2	1	9	8	7
14	15	16	17	18	10	11	12	13	5	4	3	2	1	9	8	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5
16	17	18	10	11	12	13	14	15	3	2	1	9	8	7	6	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

LI(9,9) in S4 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	10
3	4	5	6	7	8	9	1	2	12	13	14	15	16	17	18	10	11
4	5	6	7	8	9	1	2	3	13	14	15	16	17	18	10	11	12
5	6	7	8	9	1	2	3	4	14	15	16	17	18	10	11	12	13
6	7	8	9	1	2	3	4	5	15	16	17	18	10	11	12	13	14
7	8	9	1	2	3	4	5	13*	16	17	18	10	11	12	6*	14	15
8	9	1	2	3	4	5	6	7	17	18	10	11	12	13	14	15	16
9	1	2	3	4	5	13*	7	6*	18	10	11	12	8*	14	15	16	17
10	11	12	13	14	15	16	17	18	9	8	7	6	5	4	3	2	1
11	12	13	14	15	16	17	18	10	8	7	6	5	4	3	2	1	9
12	13	14	15	16	17	18	10	11	7	6	5	4	3	2	1	9	8
13	14	15	16	17	18	10	11	12	6	5	4	3	2	1	9	8	7
14	15	16	17	18	10	11	12	8*	5	4	3	2	1	9	13*	7	6
15	16	17	18	10	11	12	13	14	4	3	2	1	9	8	7	6	5

16	17	18	10	11	12	6*	14	15	3	2	1	9	13*	7	8*	5	4
17	18	10	11	12	13	14	15	16	2	1	9	8	7	6	5	4	3
18	10	11	12	13	14	15	16	17	1	9	8	7	6	5	4	3	2

# Latin Interchanges in MBC(2,11)

.....

## MBC(2,11)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	7	8	9	10	11	1	2	14	15	16	17	18	19	20	21	22	12	13
4	5	6	7	8	9	10	11	1	2	3	15	16	17	18	19	20	21	22	12	13	14
5	6	7	8	9	10	11	1	2	3	4	16	17	18	19	20	21	22	12	13	14	15
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	7	8	20	21	22	12	13	14	15	16	17	18	19
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	6	7	8	9	10	22	12	13	14	15	16	17	18	19	20	21
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	22	12	13	14	15	7	6	5	4	3	2	1	11	10	9	8
17	18	19	20	21	22	12	13	14	15	16	6	5	4	3	2	1	11	10	9	8	7
18	19	20	21	22	12	13	14	15	16	17	5	4	3	2	1	11	10	9	8	7	6
19	20	21	22	12	13	14	15	16	17	18	4	3	2	1	11	10	9	8	7	6	5
20	21	22	12	13	14	15	16	17	18	19	3	2	1	11	10	9	8	7	6	5	4
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

## Critical Set

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	**	**	**	**	**	**
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	**	**	**	**	**	**	**
3	4	5	6	7	8	9	10	11	1	2	14	15	16	**	**	**	**	**	**	**	**
4	5	6	7	8	9	10	11	1	2	3	15	16	**	**	**	**	**	**	**	**	**
5	6	7	8	9	10	11	1	2	3	4	16	**	**	**	**	**	**	**	**	**	**
6	7	8	9	10	11	1	2	3	4	5	**	**	**	**	**	**	**	**	**	**	**
7	8	9	10	11	1	2	3	4	5	**	**	**	**	**	**	**	**	**	**	**	17
8	9	10	11	1	2	3	4	5	**	**	**	**	**	**	**	**	**	**	**	**	17
9	10	11	1	2	3	4	5	**	**	**	**	**	**	**	**	**	**	**	**	17	18
10	11	1	2	3	4	5	**	**	**	9	**	**	**	**	**	**	**	**	17	18	19
11	1	2	3	4	5	**	**	**	9	10	**	**	**	**	**	**	**	17	18	19	20
12	13	14	15	16	**	**	**	**	**	**	11	10	9	8	7	**	**	**	**	**	**
13	14	15	16	**	**	**	**	**	**	**	10	9	8	7	**	**	**	**	**	**	**
14	15	16	**	**	**	**	**	**	**	**	9	8	7	**	**	**	**	**	**	**	**
15	16	**	**	**	**	**	**	**	**	**	8	7	**	**	**	**	**	**	**	**	**
16	**	**	**	**	**	**	**	**	**	**	7	**	**	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	17	**	**	**	**	**	**	**	**	**	**	6
**	**	**	**	**	**	**	**	**	**	17	18	**	**	**	**	**	**	**	**	**	6
**	**	**	**	**	**	**	**	**	**	17	18	19	**	**	**	**	**	**	**	6	5
**	**	**	**	**	**	**	**	**	**	17	18	19	20	**	**	**	**	**	**	6	5
**	**	**	**	**	**	**	**	**	**	17	18	19	20	21	**	**	**	**	**	6	5

## LI(2,6) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	22*	8	9	10	11	1	13	14	15	16	17	18	19	20	21	7*12	

3	4	5	6	7	8	9	10	11	1	2	14	15	16	17	18	19	20	21	22	12	13
4	5	6	7	8	9	10	11	1	2	3	15	16	17	18	19	20	21	22	12	13	14
5	6	7	8	9	10	11	1	2	3	4	16	17	18	19	20	21	22	12	13	14	15
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	7	8	20	21	22	12	13	14	15	16	17	18	19
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	12*	7	8	9	10	22	6*13	14	15	16	17	18	19	20	21	
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	6*12	13	14	15	7	4*	5	2*	3	11*	1	9*10	22*	8		
17	18	19	20	21	12*22*	13	14	15	16	6	5	4	3	2	1	11	10	9	8	7	
18	19	20	21	22	7*13	14	15	16	17	5	12*	3	4*	1	2*10	11*	8	9*	6		
19	20	21	22	12	13	14	15	16	17	18	4	3	2	1	11	10	9	8	7	6	5
20	21	22	12	13	14	15	16	17	18	19	3	2	1	11	10	9	8	7	6	5	4
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

LI(3,5) in S2 having pattern P2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	22*	8	9	10	11	1	2	14	15	16	17	18	19	20	21	7*12	13	
4	5	6	7	8	9	10	11	1	2	3	15	16	17	18	19	20	21	22	12	13	14
5	6	7	8	9	10	11	1	2	3	4	16	17	18	19	20	21	22	12	13	14	15
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	7	8	20	21	22	12	13	14	15	16	17	18	19
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	12*	7	8	9	10	22	6*13	14	15	16	17	18	19	20	21	
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	6	12	13	14	15	7	3*	5	4	2*10*	1	11	22*	9	8	
17	18	19	20	21	12	22	13	14	15	16	6	5	4	3	1*	2*11	10	9	8	7	
18	19	20	21	12	22	13	14	15	16	17	5	4	3	2	11*	1*10	9	8	7	6	
19	20	21	22	7	13	14	15	16	17	18	4	12*	2	1	3*11*	9	8	10*	6	5	
20	21	22	12	13	14	15	16	17	18	19	3	2	1	11	10	9	8	7	6	5	4
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

LI(4,4) in S2 having pattern P6

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	7	8	9	10	11	1	2	14	15	16	17	18	19	20	21	22	12	13
4	5	6	22*	8	9	10	11	1	2	3	15	16	17	18	19	20	21	7*12	13	14	
5	6	7	8	9	10	11	1	2	3	4	16	17	18	19	20	21	22	12	13	14	15
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	7	8	20	21	22	12	13	14	15	16	17	18	19
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	12*	7	8	9	10	22	6*13	14	15	16	17	18	19	20	21	

12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	6*12	13	14	15	7	2*	5	4	3	11*	1	22*10	9	8		
17	18	19	20	21	12*22*13	14	15	16	6	5	4	3	2	1	11	10	9	8	7		
18	19	20	21	12*22*13	14	15	16	17	5	4	3	11*	1	2*10	9	8	7	6			
19	20	21	12*22*13	14	15	16	17	18	4	3	2	1	11	10	9	8	7	6	5		
20	21	22	7*13	14	15	16	17	18	19	3	12*	1	2*10	9	8	11*	6	5	4		
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

LI(3,6) in S2 having pattern P3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	7	22*	9	10	11	1	2	14	15	16	17	18	19	20	21	8*12	13	
4	5	6	7	8	9	10	11	1	2	3	15	16	17	18	19	20	21	22	12	13	14
5	6	7	8	9	10	11	1	2	3	4	16	17	18	19	20	21	22	12	13	14	15
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	7	8	20	21	22	12	13	14	15	16	17	18	19
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	12*	7	8	9	10	22	6*13	14	15	16	17	18	19	20	21	
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	6*12	13	14	15	7	4*	5	2*	3	1*10*11	22*	9	8			
17	18	19	20	21	12*22*13	14	15	16	6	5	4	3	1*	2*11	10	9	8	7			
18	19	20	21	22	8*13	14	15	16	17	5	12*	3	4*	2*11	1*	9	10*	7	6		
19	20	21	22	12	13	14	15	16	17	18	4	3	2	1	11	10	9	8	7	6	5
20	21	22	12	13	14	15	16	17	18	19	3	2	1	11	10	9	8	7	6	5	4
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

LI(4,5) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	7	8	9	10	11	1	2	14	15	16	17	18	19	20	21	22	12	13
4	5	6	7	22*	9	10	11	1	2	3	15	16	17	18	19	20	21	8*12	13	14	
5	6	7	8	9	10	11	1	2	3	4	16	17	18	19	20	21	22	12	13	14	15
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	7	8	20	21	22	12	13	14	15	16	17	18	19
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	12*	7	8	9	10	22	6*13	14	15	16	17	18	19	20	21	
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	6*12	13	14	15	7	3*	5	4	11*	2	1	22*10	9	8		
17	18	19	20	21	12*22*13	14	15	16	6	5	4	3	2	1	11	10	9	8	7		
18	19	20	21	12*22*13	14	15	16	17	5	4	3	2	1	11	10	9	8	7	6		
19	20	21	22	8*13	14	15	16	17	18	4	12*	2	1	3*10	9	11*	7	6	5		
20	21	22	12	13	14	15	16	17	18	19	3	2	1	11	10	9	8	7	6	5	4



21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

LI(4,6) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	7	8	9	10	11	1	2	14	15	16	17	18	19	20	21	22	12	13
4	5	6	7	8	22*10	11	1	2	3	15	16	17	18	19	20	21	9*12	13	14		
5	6	7	8	9	10	11	1	2	3	4	16	17	18	19	20	21	22	12	13	14	15
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	7	8	20	21	22	12	13	14	15	16	17	18	19
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	12*	7	8	9	10	22	6*13	14	15	16	17	18	19	20	21	
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	6*12	13	14	15	7	4*	5	2*	3	11*	1	22*10	9	8		
17	18	19	20	21	12*22*13	14	15	16	6	5	4	3	2	1	11	10	9	8	7		
18	19	20	21	22	9*13	14	15	16	17	5	12*	3	4*	1	2*10	11*	8	7	6		
19	20	21	22	12	13	14	15	16	17	18	4	3	2	1	11	10	9	8	7	6	5
20	21	22	12	13	14	15	16	17	18	19	3	2	1	11	10	9	8	7	6	5	4
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

LI(5,5) in S2 having pattern P4

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	7	8	9	10	11	1	2	14	15	16	17	18	19	20	21	22	12	13
4	5	6	7	8	9	10	11	1	2	3	15	16	17	18	19	20	21	22	12	13	14
5	6	7	8	22*10	11	1	2	3	4	16	17	18	19	20	21	9*12	13	14	15		
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	7	8	20	21	22	12	13	14	15	16	17	18	19
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	12*	7	8	9	10	22	6*13	14	15	16	17	18	19	20	21	
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	6*12	13	14	15	7	3*	5	4	1*	2	22*11	10	9	8		
17	18	19	20	21	12*22*13	14	15	16	6	5	4	2*	3*	1	11	10	9	8	7		
18	19	20	21	12*22*13	14	15	16	17	5	4	3	1*	2*11	10	9	8	7	6			
19	20	21	22	9*13	14	15	16	17	18	4	12*	2	3*11	10	1*	8	7	6	5		
20	21	22	12	13	14	15	16	17	18	19	3	2	1	11	10	9	8	7	6	5	4
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

LI(5,6) in S2 having pattern P2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	7	8	9	10	11	1	2	14	15	16	17	18	19	20	21	22	12	13
4	5	6	7	8	9	10	11	1	2	3	15	16	17	18	19	20	21	22	12	13	14

5	6	7	8	9	22*11	1	2	3	4	16	17	18	19	20	21	10*12	13	14	15		
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	7	8	20	21	22	12	13	14	15	16	17	18	19
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	12*	7	8	9	10	22	6*13	14	15	16	17	18	19	20	21	
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	6*12	13	14	15	7	4*	5	3*	1*	2	22*11	10	9	8	7	
17	18	19	20	21	12*22*13	14	15	16	6	5	4	2*	3*	1	11	10	9	8	7	6	
18	19	20	21	22	10*13	14	15	16	17	5	12*	3	4*	2*11	1*	9	8	7	6	5	
19	20	21	22	12	13	14	15	16	17	18	4	3	2	1	11	10	9	8	7	6	5
20	21	22	12	13	14	15	16	17	18	19	3	2	1	11	10	9	8	7	6	5	4
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

LI(6,6) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	7	8	9	10	11	1	2	14	15	16	17	18	19	20	21	22	12	13
4	5	6	7	8	9	10	11	1	2	3	15	16	17	18	19	20	21	22	12	13	14
5	6	7	8	9	10	11	1	2	3	4	16	17	18	19	20	21	22	12	13	14	15
6	7	8	9	10	22*	1	2	3	4	5	17	18	19	20	21	11*12	13	14	15	16	
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	7	8	20	21	22	12	13	14	15	16	17	18	19
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	12*	7	8	9	10	22	6*13	14	15	16	17	18	19	20	21	
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	6*12	13	14	15	7	4*	5	2*	3	22*	1	11	10	9	8	7
17	18	19	20	21	12*22*13	14	15	16	6	5	4	3	2	1	11	10	9	8	7	6	5
18	19	20	21	22	11*13	14	15	16	17	5	12*	3	4*	1	2*10	9	8	7	6	5	4
19	20	21	22	12	13	14	15	16	17	18	4	3	2	1	11	10	9	8	7	6	5
20	21	22	12	13	14	15	16	17	18	19	3	2	1	11	10	9	8	7	6	5	4
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

LI(10,11) in S4 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	7	8	9	10	11	1	2	14	15	16	17	18	19	20	21	22	12	13
4	5	6	7	8	9	10	11	1	2	3	15	16	17	18	19	20	21	22	12	13	14
5	6	7	8	9	10	11	1	2	3	4	16	17	18	19	20	21	22	12	13	14	15
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	16*19	20	21	22	12	13	14	15	7*17	18		
9	10	11	1	2	3	4	5	6	7	8	20	21	22	12	13	14	15	16	17	18	19
10	11	1	2	3	4	5	6	16*	8	7*21	22	12	13	14	15	9*17	18	19	20		
11	1	2	3	4	5	6	7	8	9	10	22	12	13	14	15	16	17	18	19	20	21
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10

15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	22	12	13	14	15	7	6	5	4	3	2	1	11	10	9	8
17	18	19	20	21	22	12	13	14	15	9*	6	5	4	3	2	1	11	10	16*	8	7
18	19	20	21	22	12	13	14	15	16	17	5	4	3	2	1	11	10	9	8	7	6
19	20	21	22	12	13	14	15	7*17	18	4	3	2	1	11	10	16*	8	9*	6	5	4
20	21	22	12	13	14	15	16	17	18	19	3	2	1	11	10	9	8	7	6	5	4
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

LI(11,10) in S4 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	7	8	9	10	11	1	2	14	15	16	17	18	19	20	21	22	12	13
4	5	6	7	8	9	10	11	1	2	3	15	16	17	18	19	20	21	22	12	13	14
5	6	7	8	9	10	11	1	2	3	4	16	17	18	19	20	21	22	12	13	14	15
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	16*	8	20	21	22	12	13	14	15	7*17	18	19	20
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	6	16*	8	7*10	22	12	13	14	15	9*17	18	19	20	21	22	12
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	22	12	13	14	15	7	6	5	4	3	2	1	11	10	9	8
17	18	19	20	21	22	12	13	14	15	16	6	5	4	3	2	1	11	10	9	8	7
18	19	20	21	22	12	13	14	15	9*17	5	4	3	2	1	11	10	16*	8	7	6	5
19	20	21	22	12	13	14	15	16	17	18	4	3	2	1	11	10	9	8	7	6	5
20	21	22	12	13	14	15	7*17	18	19	3	2	1	11	10	16*	8	9*	6	5	4	3
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

LI(11,11) in S4 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	3	4	5	6	7	8	9	10	11	1	13	14	15	16	17	18	19	20	21	22	12
3	4	5	6	7	8	9	10	11	1	2	14	15	16	17	18	19	20	21	22	12	13
4	5	6	7	8	9	10	11	1	2	3	15	16	17	18	19	20	21	22	12	13	14
5	6	7	8	9	10	11	1	2	3	4	16	17	18	19	20	21	22	12	13	14	15
6	7	8	9	10	11	1	2	3	4	5	17	18	19	20	21	22	12	13	14	15	16
7	8	9	10	11	1	2	3	4	5	6	18	19	20	21	22	12	13	14	15	16	17
8	9	10	11	1	2	3	4	5	6	7	19	20	21	22	12	13	14	15	16	17	18
9	10	11	1	2	3	4	5	6	7	16*20	21	22	12	13	14	15	8*17	18	19	20	21
10	11	1	2	3	4	5	6	7	8	9	21	22	12	13	14	15	16	17	18	19	20
11	1	2	3	4	5	6	7	16*	9	8*22	12	13	14	15	10*17	18	19	20	21	22	12
12	13	14	15	16	17	18	19	20	21	22	11	10	9	8	7	6	5	4	3	2	1
13	14	15	16	17	18	19	20	21	22	12	10	9	8	7	6	5	4	3	2	1	11
14	15	16	17	18	19	20	21	22	12	13	9	8	7	6	5	4	3	2	1	11	10
15	16	17	18	19	20	21	22	12	13	14	8	7	6	5	4	3	2	1	11	10	9
16	17	18	19	20	21	22	12	13	14	15	7	6	5	4	3	2	1	11	10	9	8
17	18	19	20	21	22	12	13	14	15	10*	6	5	4	3	2	1	11	16*	9	8	7
18	19	20	21	22	12	13	14	15	16	17	5	4	3	2	1	11	10	9	8	7	6
19	20	21	22	12	13	14	15	8*17	18	4	3	2	1	11	16*	9	10*	7	6	5	4
20	21	22	12	13	14	15	16	17	18	19	3	2	1	11	10	9	8	7	6	5	4
21	22	12	13	14	15	16	17	18	19	20	2	1	11	10	9	8	7	6	5	4	3
22	12	13	14	15	16	17	18	19	20	21	1	11	10	9	8	7	6	5	4	3	2

Latin Interchanges for mbc(2,13)  
.....

MBC(2,13)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	7	8	9	10	11	12	26	14	15	16	17	18	19	20	21	22	23	24	25
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	26	14	15	16	17	18	8	7	6	5	4	3	2	1	13	12	11	10	9
20	21	22	23	24	25	26	14	15	16	17	18	19	7	6	5	4	3	2	1	13	12	11	10	9	8
21	22	23	24	25	26	14	15	16	17	18	19	20	6	5	4	3	2	1	13	12	11	10	9	8	7
22	23	24	25	26	14	15	16	17	18	19	20	21	5	4	3	2	1	13	12	11	10	9	8	7	6
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

Critical set

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	**	**	**	**	**	**	**
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	**	**	**	**	**	**	**	**
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	**	**	**	**	**	**	**	**	**
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	**	**	**	**	**	**	**	**	**	**
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	**	**	**	**	**	**	**	**	**	**	**
6	7	8	9	10	11	12	13	1	2	3	4	5	19	**	**	**	**	**	**	**	**	**	**	**	**
7	8	9	10	11	12	13	1	2	3	4	5	6	**	**	**	**	**	**	**	**	**	**	**	**	**
8	9	10	11	12	13	1	2	3	4	5	6	**	**	**	**	**	**	**	**	**	**	**	**	**	20
9	10	11	12	13	1	2	3	4	5	6	**	**	**	**	**	**	**	**	**	**	**	**	**	**	20
10	11	12	13	1	2	3	4	5	6	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	20
11	12	13	1	2	3	4	5	6	**	**	**	10	**	**	**	**	**	**	**	**	**	**	**	**	20
12	13	1	2	3	4	5	6	**	**	**	10	11	**	**	**	**	**	**	**	**	**	**	**	**	20
13	1	2	3	4	5	6	**	**	**	10	11	12	**	**	**	**	**	**	**	**	**	**	**	**	20
14	15	16	17	18	19	**	**	**	**	**	**	**	13	12	11	10	9	8	**	**	**	**	**	**	**
15	16	17	18	19	**	**	**	**	**	**	**	**	12	11	10	9	8	**	**	**	**	**	**	**	**
16	17	18	19	**	**	**	**	**	**	**	**	**	11	10	9	8	**	**	**	**	**	**	**	**	**
17	18	19	**	**	**	**	**	**	**	**	**	**	10	9	8	**	**	**	**	**	**	**	**	**	**
18	19	**	**	**	**	**	**	**	**	**	**	**	9	8	**	**	**	**	**	**	**	**	**	**	**
19	**	**	**	**	**	**	**	**	**	**	**	**	8	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	**	**	**	**	**	**	**	**	**	**	**	20	**	**	**	**	**	**	**	**	**	**	**	7
**	**	**	**	**	**	**	**	**	**	**	**	**	20	21	**	**	**	**	**	**	**	**	**	**	7
**	**	**	**	**	**	**	**	**	**	**	**	**	20	21	22	**	**	**	**	**	**	**	**	**	7
**	**	**	**	**	**	**	**	**	**	**	**	**	20	21	22	23	**	**	**	**	**	**	**	**	7
**	**	**	**	**	**	**	**	**	**	**	**	**	20	21	22	23	24	**	**	**	**	**	**	**	7
**	**	**	**	**	**	**	**	**	**	**	**	**	20	21	22	23	24	25	**	**	**	**	**	**	7

LI(2,7) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	26*	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	8*14	
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*15	16	17	18	19	20	21	22	23	24	25	
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*14	15	16	17	18	8	5*	6	3*	4	1*	2	12*13	10*11	26*	9			
20	21	22	23	24	25	14*26*	15	16	17	18	19	7	6	5	4	3	2	1	13	12	11	10	9	8	
21	22	23	24	25	26	8*15	16	17	18	19	20	6	14*	4	5*	2	3*13	1*11	12*	9	10*	7			
22	23	24	25	26	14	15	16	17	18	19	20	21	5	4	3	2	1	13	12	11	10	9	8	7	6
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(3,6) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	26*	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	8*14	15	
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*15	16	17	18	19	20	21	22	23	24	25	
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*14	15	16	17	18	8	4*	6	5	1*	3	2	11*13	12	26*10	9			
20	21	22	23	24	25	14*26*	15	16	17	18	19	7	6	5	4	3	2	1	13	12	11	10	9	8	
21	22	23	24	25	14*26*	15	16	17	18	19	20	6	5	4	3	2	1	13	12	11	10	9	8	7	
22	23	24	25	26	8*15	16	17	18	19	20	21	5	14*	3	2	4*13	12	1*10	9	11*	7	6			
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(4,5) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	26*	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	8*14	15	16	
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*15	16	17	18	19	20	21	22	23	24	25	
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*14	15	16	17	18	8	3*	6	5	4	12*	2	1	13	26*11	10	9	8	
20	21	22	23	24	25	14*26*	15	16	17	18	19	7	6	5	4	3	2	1	13	12	11	10	9	8	
21	22	23	24	25	14*26*	15	16	17	18	19	20	6	5	4	3	2	1	13	12	11	10	9	8	7	
22	23	24	25	14*26*	15	16	17	18	19	20	21	5	4	3	2	1	13	12	11	10	9	8	7	6	
23	24	25	26	8*15	16	17	18	19	20	21	22	4	14*	2	1	13	3*11	10	9	12*	7	6	5	4	
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(3,7) in S2 having pattern P2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	26*10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	9*14	15		
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*15	16	17	18	19	20	21	22	23	24	25	
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*14	15	16	17	18	8	5*	6	3*	4	2*13*	1	11*12	26*10	9				
20	21	22	23	24	25	14*26*	15	16	17	18	19	7	6	5	4	3	1*	2*13	12	11	10	9	8		
21	22	23	24	25	26	9*15	16	17	18	19	20	6	14*	4	5*	2	3*	1*12	13*10	11*	8	7			
22	23	24	25	26	14	15	16	17	18	19	20	5	4	3	2	1	13	12	11	10	9	8	7	6	
23	24	25	26	14	15	16	17	18	19	20	21	4	3	2	1	13	12	11	10	9	8	7	6	5	
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(4,6) in S2 having pattern P7

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	26*10	11	12	13	1	2	3	4	17	18	19	20	21	22	23	24	25	9*14	15	16	17
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*15	16	17	18	19	20	21	22	23	24	25	26
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*14	15	16	17	18	8	4*	6	5	2*	3	12*	1	13	26*11	10	9	8	7
20	21	22	23	24	25	14*26*15	16	17	18	19	7	6	5	4	3	1*	2*13	12	11	10	9	8	7	6	5
21	22	23	24	25	14*26*15	16	17	18	19	20	6	5	4	3	1*	2*13	12	11	10	9	8	7	6	5	4
22	23	24	25	26	9*15	16	17	18	19	20	21	5	14*	3	2	4*13	1*11	10	12*	8	7	6	5	4	3
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(5,5) in S2 having pattern P4

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	26*10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	9*14	15	16	17	18	19
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*15	16	17	18	19	20	21	22	23	24	25	26
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*14	15	16	17	18	8	3*	6	5	4	13*	2	1	26*12	11	10	9	8	7
20	21	22	23	24	25	14*26*15	16	17	18	19	7	6	5	4	2*	3*	1	13	12	11	10	9	8	7	6
21	22	23	24	25	14*26*15	16	17	18	19	20	6	5	4	3	1*	2*13	12	11	10	9	8	7	6	5	4
22	23	24	25	14*26*15	16	17	18	19	20	21	5	4	3	2	13*	1*12	11	10	9	8	7	6	5	4	3
23	24	25	26	9*15	16	17	18	19	20	21	22	4	14*	2	1	3*12	11	10	13*	8	7	6	5	4	3
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(4,7) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	26*	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	10*	14	15	16
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*	15	16	17	18	19	20	21	22	23	24	25
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*	14	15	16	17	18	8	5*	6	3*	4	1*	2	12*	13	26*	11	10	9
20	21	22	23	24	25	14*	26*	15	16	17	18	19	7	6	5	4	3	2	1	13	12	11	10	9	8
21	22	23	24	25	26	10*	15	16	17	18	19	20	6	14*	4	5*	2	3*	13	1*	11	12*	9	8	7
22	23	24	25	26	14	15	16	17	18	19	20	21	5	4	3	2	1	13	12	11	10	9	8	7	6
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(5,6) in S2 having pattern P2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	26*	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	10*	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*	15	16	17	18	19	20	21	22	23	24	25
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*	14	15	16	17	18	8	4*	6	5	3*	13*	2	1	26*	12	11	10	9
20	21	22	23	24	25	14*	26*	15	16	17	18	19	7	6	5	4	2*	3*	1	13	12	11	10	9	8
21	22	23	24	25	14*	26*	15	16	17	18	19	20	6	5	4	3	1*	2*	13	12	11	10	9	8	7
22	23	24	25	26	10*	15	16	17	18	19	20	21	5	14*	3	2	4*	1*	12	11	13*	9	8	7	6
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(5,7) in S2 having pattern P3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14



3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	10	26*12	13	1	2	3	4	18	19	20	21	22	23	24	25	11*14	15	16	17	18	19
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*15	16	17	18	19	20	21	22	23	24	25	26
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*14	15	16	17	18	8	5*	6	3*	4	2*13*	1	26*12	11	10	9	8	7	6
20	21	22	23	24	25	14*26*15	16	17	18	19	7	6	5	4	2*	3*	1	13	12	11	10	9	8	7	6
21	22	23	24	25	26	11*15	16	17	18	19	20	6	14*	4	5*	3*	1	2*12	13*10	9	8	7	6	5	4
22	23	24	25	26	14	15	16	17	18	19	20	21	5	4	3	2	1	13	12	11	10	9	8	7	6
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(6,6) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	26*12	13	1	2	3	4	5	19	20	21	22	23	24	25	11*14	15	16	17	18	19	20
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*15	16	17	18	19	20	21	22	23	24	25	26
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*14	15	16	17	18	8	4*	6	5	1*	3	2	26*13	12	11	10	9	8	7
20	21	22	23	24	25	14*26*15	16	17	18	19	7	6	5	4	3	2	1	13	12	11	10	9	8	7	6
21	22	23	24	25	14*26*15	16	17	18	19	20	6	5	4	3	2	1	13	12	11	10	9	8	7	6	5
22	23	24	25	26	11*15	16	17	18	19	20	21	5	14*	3	2	4*13	12	1*	10	9	8	7	6	5	4
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(6,7) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16

5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	26*	13	1	2	3	4	5	19	20	21	22	23	24	25	12*	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*	15	16	17	18	19	20	21	22	23	24	25
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*	14	15	16	17	18	8	5*	6	3*	4	1*	2	26*	13	12	11	10	9
20	21	22	23	24	25	14*	26*	15	16	17	18	19	7	6	5	4	3	2	1	13	12	11	10	9	8
21	22	23	24	25	26	12*	15	16	17	18	19	20	6	14*	4	5*	2	3*	13	1*	11	10	9	8	7
22	23	24	25	26	14	15	16	17	18	19	20	21	5	4	3	2	1	13	12	11	10	9	8	7	6
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(7,7) in S2 having pattern P2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	26*	1	2	3	4	5	6	20	21	22	23	24	25	13*	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	14*	8	9	10	11	12	26	7*	15	16	17	18	19	20	21	22	23	24	25
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	7*	14	15	16	17	18	8	5*	6	4*	2*	3	26*	1	13	12	11	10	9
20	21	22	23	24	25	14*	26*	15	16	17	18	19	7	6	5	3*	4*	2	1	13	12	11	10	9	8
21	22	23	24	25	26	13*	15	16	17	18	19	20	6	14*	4	5*	3*	1	2*	12	11	10	9	8	7
22	23	24	25	26	14	15	16	17	18	19	20	21	5	4	3	2	1	13	12	11	10	9	8	7	6
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(11,13) in S4 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17



8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	19*	9	10	24	25	26	14	15	16	17	18	8*20	21	22	23	24
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	7	19*	9	8*11	12	26	14	15	16	17	18	10*20	21	22	23	24	25	26	14
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	26	14	15	16	17	18	8	7	6	5	4	3	2	1	13	12	11	10	9
20	21	22	23	24	25	26	14	15	16	17	18	19	7	6	5	4	3	2	1	13	12	11	10	9	8
21	22	23	24	25	26	14	15	16	17	18	19	20	6	5	4	3	2	1	13	12	11	10	9	8	7
22	23	24	25	26	14	15	16	17	18	10*20	21	5	4	3	2	1	13	12	11	19*	9	8	7	6	5
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	8*20	21	22	23	3	2	1	13	12	11	19*	9	10*	7	6	5	4	3
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(12,13) in S4 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	19*23	24	25	26	14	15	16	17	18	9*20	21	22	23	24
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	19*10	9*25	26	14	15	16	17	18	11*20	21	22	23	24	25	26	14
13	1	2	3	4	5	6	7	8	9	10	11	12	26	14	15	16	17	18	19	20	21	22	23	24	25
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	26	14	15	16	17	18	8	7	6	5	4	3	2	1	13	12	11	10	9
20	21	22	23	24	25	26	14	15	16	17	18	11*	7	6	5	4	3	2	1	13	12	19*10	9	8	7
21	22	23	24	25	26	14	15	16	17	18	19	20	6	5	4	3	2	1	13	12	11	10	9	8	7
22	23	24	25	26	14	15	16	17	18	9*20	21	5	4	3	2	1	13	12	19*10	11*	8	7	6	5	4
23	24	25	26	14	15	16	17	18	19	20	21	22	4	3	2	1	13	12	11	10	9	8	7	6	5
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(13,12) in S4 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21

10	11	12	13	1	2	3	4	5	6	7	8	9	23	24	25	26	14	15	16	17	18	19	20	21	22
11	12	13	1	2	3	4	5	6	7	8	19*10	24	25	26	14	15	16	17	18	9*20	21	22	23	24	
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	7	8	19*10	9*12	26	14	15	16	17	18	11*20	21	22	23	24	25			
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	26	14	15	16	17	18	8	7	6	5	4	3	2	1	13	12	11	10	9
20	21	22	23	24	25	26	14	15	16	17	18	19	7	6	5	4	3	2	1	13	12	11	10	9	8
21	22	23	24	25	26	14	15	16	17	18	11*20	6	5	4	3	2	1	13	12	19*10	9	8	7		
22	23	24	25	26	14	15	16	17	18	19	20	21	5	4	3	2	1	13	12	11	10	9	8	7	6
23	24	25	26	14	15	16	17	18	9*20	21	22	4	3	2	1	13	12	19*10	11*	8	7	6	5		
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

LI(13,13) in S4 having pattern P2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	3	4	5	6	7	8	9	10	11	12	13	1	15	16	17	18	19	20	21	22	23	24	25	26	14
3	4	5	6	7	8	9	10	11	12	13	1	2	16	17	18	19	20	21	22	23	24	25	26	14	15
4	5	6	7	8	9	10	11	12	13	1	2	3	17	18	19	20	21	22	23	24	25	26	14	15	16
5	6	7	8	9	10	11	12	13	1	2	3	4	18	19	20	21	22	23	24	25	26	14	15	16	17
6	7	8	9	10	11	12	13	1	2	3	4	5	19	20	21	22	23	24	25	26	14	15	16	17	18
7	8	9	10	11	12	13	1	2	3	4	5	6	20	21	22	23	24	25	26	14	15	16	17	18	19
8	9	10	11	12	13	1	2	3	4	5	6	7	21	22	23	24	25	26	14	15	16	17	18	19	20
9	10	11	12	13	1	2	3	4	5	6	7	8	22	23	24	25	26	14	15	16	17	18	19	20	21
10	11	12	13	1	2	3	4	5	6	7	8	19*23	24	25	26	14	15	16	17	18	9*20	21	22		
11	12	13	1	2	3	4	5	6	7	8	9	10	24	25	26	14	15	16	17	18	19	20	21	22	23
12	13	1	2	3	4	5	6	7	8	9	10	11	25	26	14	15	16	17	18	19	20	21	22	23	24
13	1	2	3	4	5	6	7	8	19*10	11	9*26	14	15	16	17	18	12*20	21	22	23	24	25			
14	15	16	17	18	19	20	21	22	23	24	25	26	13	12	11	10	9	8	7	6	5	4	3	2	1
15	16	17	18	19	20	21	22	23	24	25	26	14	12	11	10	9	8	7	6	5	4	3	2	1	13
16	17	18	19	20	21	22	23	24	25	26	14	15	11	10	9	8	7	6	5	4	3	2	1	13	12
17	18	19	20	21	22	23	24	25	26	14	15	16	10	9	8	7	6	5	4	3	2	1	13	12	11
18	19	20	21	22	23	24	25	26	14	15	16	17	9	8	7	6	5	4	3	2	1	13	12	11	10
19	20	21	22	23	24	25	26	14	15	16	17	18	8	7	6	5	4	3	2	1	13	12	11	10	9
20	21	22	23	24	25	26	14	15	16	17	18	12*	7	6	5	4	3	2	1	13	11*19*10	9	8		
21	22	23	24	25	26	14	15	16	17	18	19	20	6	5	4	3	2	1	13	10*12*11*	9	8	7		
22	23	24	25	26	14	15	16	17	18	19	20	21	5	4	3	2	1	13	11*12*	9*10*	8	7	6		
23	24	25	26	14	15	16	17	18	9*20	21	22	4	3	2	1	13	12	19*11*10*	8	7	6	5			
24	25	26	14	15	16	17	18	19	20	21	22	23	3	2	1	13	12	11	10	9	8	7	6	5	4
25	26	14	15	16	17	18	19	20	21	22	23	24	2	1	13	12	11	10	9	8	7	6	5	4	3
26	14	15	16	17	18	19	20	21	22	23	24	25	1	13	12	11	10	9	8	7	6	5	4	3	2

.....

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	16
3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	18	19	20	21	22	23	24	25	26	27	28	29	30	16	17
4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	19	20	21	22	23	24	25	26	27	28	29	30	16	17	18
5	6	7	8	9	10	11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	30	16	17	18	19
6	7	8	9	10	11	12	13	14	15	1	2	3	4	5	21	22	23	24	25	26	27	28	29	30	16	17	18	19	20
7	8	9	10	11	12	13	14	15	1	2	3	4	5	6	22	23	24	25	26	27	28	29	30	16	17	18	19	20	21
8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	23	24	25	26	27	28	29	30	16	17	18	19	20	21	22
9	10	11	12	13	14	15	1	2	3	4	5	6	7	8	24	25	26	27	28	29	30	16	17	18	19	20	21	22	23
10	11	12	13	14	15	1	2	3	4	5	6	7	8	9	25	26	27	28	29	30	16	17	18	19	20	21	22	23	24
11	12	13	14	15	1	2	3	4	5	6	7	8	9	10	26	27	28	29	30	16	17	18	19	20	21	22	23	24	25
12	13	14	15	1	2	3	4	5	6	7	8	9	10	11	27	28	29	30	16	17	18	19	20	21	22	23	24	25	26
13	14	15	1	2	3	4	5	6	7	8	9	10	11	12	28	29	30	16	17	18	19	20	21	22	23	24	25	26	27
14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	29	30	16	17	18	19	20	21	22	23	24	25	26	27	28
15	1	2	3	4	5	6	7	8	9	10	11	12	13	14	30	16	17	18	19	20	21	22	23	24	25	26	27	28	29
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	1

[illegible]



15	1	2	3	4	5	6	7	16*	9	10	11	12	13	14	30	8*17	18	19	20	21	22	23	24	25	26	27	28	29	
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	12
21	22	23	24	25	26	27	28	29	30	16	17	18	19	20	10	9	8	7	6	5	4	3	2	1	15	14	13	12	11
22	23	24	25	26	27	28	29	8*16	17	18	19	20	21	9	5*	7	6	2*	4	3	15*	1	12*14	13	30*11	10			
23	24	25	26	27	28	29	16*30*17	18	19	20	21	22	8	7	6	5	4	3	1*	2*15	14	13	12	11	10	9			
24	25	26	27	28	29	16*30*17	18	19	20	21	22	23	7	6	5	4	3	2	15*	1*14	13	12	11	10	9	8			
25	26	27	28	29	30	9*17	18	19	20	21	22	23	24	6	16*	4	3	5*	1	2*14	13	15*11	10	12*	8	7			
26	27	28	29	30	16	17	18	19	20	21	22	23	24	25	5	4	3	2	1	15	14	13	12	11	10	9	8	7	6
27	28	29	30	16	17	18	19	20	21	22	23	24	25	26	4	3	2	1	15	14	13	12	11	10	9	8	7	6	5
28	29	30	16	17	18	19	20	21	22	23	24	25	26	27	3	2	1	15	14	13	12	11	10	9	8	7	6	5	4
29	30	16	17	18	19	20	21	22	23	24	25	26	27	28	2	1	15	14	13	12	11	10	9	8	7	6	5	4	3
30	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	15	14	13	12	11	10	9	8	7	6	5	4	3	2

LI(4,6) in S2 having pattern P5

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	16
3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	18	19	20	21	22	23	24	25	26	27	28	29	30	16	17
4	5	6	7	8	30*10	11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	9*16	17	18		
5	6	7	8	9	10	11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	30	16	17	18	19
6	7	8	9	10	11	12	13	14	15	1	2	3	4	5	21	22	23	24	25	26	27	28	29	30	16	17	18	19	20
7	8	9	10	11	12	13	14	15	1	2	3	4	5	6	22	23	24	25	26	27	28	29	30	16	17	18	19	20	21
8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	23	24	25	26	27	28	29	30	16	17	18	19	20	21	22
9	10	11	12	13	14	15	1	2	3	4	5	6	7	8	24	25	26	27	28	29	30	16	17	18	19	20	21	22	23
10	11	12	13	14	15	1	2	3	4	5	6	7	8	9	25	26	27	28	29	30	16	17	18	19	20	21	22	23	24
11	12	13	14	15	1	2	3	4	5	6	7	8	9	10	26	27	28	29	30	16	17	18	19	20	21	22	23	24	25
12	13	14	15	1	2	3	4	5	6	7	8	9	10	11	27	28	29	30	16	17	18	19	20	21	22	23	24	25	26
13	14	15	1	2	3	4	5	6	7	8	9	10	11	12	28	29	30	16	17	18	19	20	21	22	23	24	25	26	27
14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	29	30	16	17	18	19	20	21	22	23	24	25	26	27	28
15	1	2	3	4	5	6	7	16*	9	10	11	12	13	14	30	8*17	18	19	20	21	22	23	24	25	26	27	28	29	
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	12
21	22	23	24	25	26	27	28	29	30	16	17	18	19	20	10	9	8	7	6	5	4	3	2	1	15	14	13	12	11
22	23	24	25	26	27	28	29	8*16	17	18	19	20	21	9	4*	7	6	5	2*	3	13*	1	15	14	30*12	11	10		
23	24	25	26	27	28	29	16*30*17	18	19	20	21	22	8	7	6	5	4	3	2	1	15	14	13	12	11	10	9		
24	25	26	27	28	29	16*30*17	18	19	20	21	22	23	7	6	5	4	3	15*	1	2*14	13	12	11	10	9	8			
25	26	27	28	29	30	9*17	18	19	20	21	22	23	24	6	5	4	3	2	1	15	14	13	12	11	10	9	8	7	
26	27	28	29	30	16	17	18	19	20	21	22	23	24	25	5	16*	3	2	1	4*14	15*12	11	10	13*	8	7	6		
27	28	29	30	16	17	18	19	20	21	22	23	24	25	26	4	3	2	1	15	14	13	12	11	10	9	8	7	6	5
28	29	30	16	17	18	19	20	21	22	23	24	25	26	27	3	2	1	15	14	13	12	11	10	9	8	7	6	5	4
29	30	16	17	18	19	20	21	22	23	24	25	26	27	28	2	1	15	14	13	12	11	10	9	8	7	6	5	4	3
30	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	15	14	13	12	11	10	9	8	7	6	5	4	3	2

LI(5,5) in S2 having pattern P4

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	16
3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	18	19	20	21	22	23	24	25	26	27	28	29	30	16	17
4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	19	20	21	22	23	24	25	26	27	28	29	30	16	17	18
5	6	7	8	30*10	11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	9*16	17	18	19		
6	7	8	9	10	11	12	13	14	15	1	2	3	4	5	21	22	23	24	25	26	27	28	29	30	16	17	18	19	20
7	8	9	10	11	12	13	14	15	1	2	3	4	5	6	22	23	24	25	26	27	28	29	30	16	17	18	19	20	21
8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	23	24	25	26	27	28	29	30	16	17	18	19	20	21	22



9	10	11	12	13	14	15	1	2	3	4	5	6	7	8	24	25	26	27	28	29	30	16	17	18	19	20	21	22	23
10	11	12	13	14	15	1	2	3	4	5	6	7	8	9	25	26	27	28	29	30	16	17	18	19	20	21	22	23	24
11	12	13	14	15	1	2	3	4	5	6	7	8	9	10	26	27	28	29	30	16	17	18	19	20	21	22	23	24	25
12	13	14	15	1	2	3	4	5	6	7	8	9	10	11	27	28	29	30	16	17	18	19	20	21	22	23	24	25	26
13	14	15	1	2	3	4	5	6	7	8	9	10	11	12	28	29	30	16	17	18	19	20	21	22	23	24	25	26	27
14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	29	30	16	17	18	19	20	21	22	23	24	25	26	27	28
15	1	2	3	4	5	6	7	16*	9	10	11	12	13	14	30	8*	17	18	19	20	21	22	23	24	25	26	27	28	29
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	12
21	22	23	24	25	26	27	28	29	30	16	17	18	19	20	10	9	8	7	6	5	4	3	2	1	15	14	13	12	11
22	23	24	25	26	27	28	29	8*16	17	18	19	20	21	9	3*	7	6	5	4	14*	2	1	15	30*13	12	11	10		
23	24	25	26	27	28	29	16*30*17	18	19	20	21	22	8	7	6	5	4	2*	3*	1	15	14	13	12	11	10	9		
24	25	26	27	28	29	16*30*17	18	19	20	21	22	23	7	6	5	4	3	1*	2*15	14	13	12	11	10	9	8			
25	26	27	28	29	16*30*17	18	19	20	21	22	23	24	6	5	4	3	2	15*	1*14	13	12	11	10	9	8	7			
26	27	28	29	16*30*17	18	19	20	21	22	23	24	25	5	4	3	2	1	14*15*13	12	11	10	9	8	7	6				
27	28	29	30	9*17	18	19	20	21	22	23	24	25	26	4	16*	2	1	15	3*13	12	11	10	14*	8	7	6	5		
28	29	30	16	17	18	19	20	21	22	23	24	25	26	27	3	2	1	15	14	13	12	11	10	9	8	7	6	5	4
29	30	16	17	18	19	20	21	22	23	24	25	26	27	28	2	1	15	14	13	12	11	10	9	8	7	6	5	4	3
30	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	15	14	13	12	11	10	9	8	7	6	5	4	3	2

LI(3,8) in S2 having pattern P3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	16
3	4	5	6	7	8	9	30*11	12	13	14	15	1	2	18	19	20	21	22	23	24	25	26	27	28	29	10*16	17		
4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	19	20	21	22	23	24	25	26	27	28	29	30	16	17	18
5	6	7	8	9	10	11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	30	16	17	18	19
6	7	8	9	10	11	12	13	14	15	1	2	3	4	5	21	22	23	24	25	26	27	28	29	30	16	17	18	19	20
7	8	9	10	11	12	13	14	15	1	2	3	4	5	6	22	23	24	25	26	27	28	29	30	16	17	18	19	20	21
8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	23	24	25	26	27	28	29	30	16	17	18	19	20	21	22
9	10	11	12	13	14	15	1	2	3	4	5	6	7	8	24	25	26	27	28	29	30	16	17	18	19	20	21	22	23
10	11	12	13	14	15	1	2	3	4	5	6	7	8	9	25	26	27	28	29	30	16	17	18	19	20	21	22	23	24
11	12	13	14	15	1	2	3	4	5	6	7	8	9	10	26	27	28	29	30	16	17	18	19	20	21	22	23	24	25
12	13	14	15	1	2	3	4	5	6	7	8	9	10	11	27	28	29	30	16	17	18	19	20	21	22	23	24	25	26
13	14	15	1	2	3	4	5	6	7	8	9	10	11	12	28	29	30	16	17	18	19	20	21	22	23	24	25	26	27
14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	29	30	16	17	18	19	20	21	22	23	24	25	26	27	28
15	1	2	3	4	5	6	7	16*	9	10	11	12	13	14	30	8*17	18	19	20	21	22	23	24	25	26	27	28	29	
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	12
21	22	23	24	25	26	27	28	29	30	16	17	18	19	20	10	9	8	7	6	5	4	3	2	1	15	14	13	12	11
22	23	24	25	26	27	28	29	8*16	17	18	19	20	21	9	6*	7	4*	5	2*	3	1*14*15	12*13	30*11	10					
23	24	25	26	27	28	29	16*30*17	18	19	20	21	22	8	7	6	5	4	3	1*	2*15	14	13	12	11	10	9			
24	25	26	27	28	29	30	10*17	18	19	20	21	22	23	7	16*	5	6*	3	4*	2*15	1*13	14*11	12*	9	8				
25	26	27	28	29	30	16	17	18	19	20	21	22	23	24	6	5	4	3	2	1	15	14	13	12	11	10	9	8	7
26	27	28	29	30	16	17	18	19	20	21	22	23	24	25	5	4	3	2	1	15	14	13	12	11	10	9	8	7	6
27	28	29	30	16	17	18	19	20	21	22	23	24	25	26	4	3	2	1	15	14	13	12	11	10	9	8	7	6	5
28	29	30	16	17	18	19	20	21	22	23	24	25	26	27	3	2	1	15	14	13	12	11	10	9	8	7	6	5	4
29	30	16	17	18	19	20	21	22	23	24	25	26	27	28	2	1	15	14	13	12	11	10	9	8	7	6	5	4	3
30	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	15	14	13	12	11	10	9	8	7	6	5	4	3	2

LI(4,7) in S2 having pattern P3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	16

3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	18	19	20	21	22	23	24	25	26	27	28	29	30	16	17
4	5	6	7	8	9	30*	11	12	13	14	15	1	2	3	19	20	21	22	23	24	25	26	27	28	29	10*	16	17	18
5	6	7	8	9	10	11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	10*	16	17	18	
6	7	8	9	10	11	12	13	14	15	1	2	3	4	5	21	22	23	24	25	26	27	28	29	30	16	17	18	19	20
7	8	9	10	11	12	13	14	15	1	2	3	4	5	6	22	23	24	25	26	27	28	29	30	16	17	18	19	20	21
8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	23	24	25	26	27	28	29	30	16	17	18	19	20	21	22
9	10	11	12	13	14	15	1	2	3	4	5	6	7	8	24	25	26	27	28	29	30	16	17	18	19	20	21	22	23
10	11	12	13	14	15	1	2	3	4	5	6	7	8	9	25	26	27	28	29	30	16	17	18	19	20	21	22	23	24
11	12	13	14	15	1	2	3	4	5	6	7	8	9	10	26	27	28	29	30	16	17	18	19	20	21	22	23	24	25
12	13	14	15	1	2	3	4	5	6	7	8	9	10	11	27	28	29	30	16	17	18	19	20	21	22	23	24	25	26
13	14	15	1	2	3	4	5	6	7	8	9	10	11	12	28	29	30	16	17	18	19	20	21	22	23	24	25	26	27
14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	29	30	16	17	18	19	20	21	22	23	24	25	26	27	28
15	1	2	3	4	5	6	7	16*	9	10	11	12	13	14	30	8*	17	18	19	20	21	22	23	24	25	26	27	28	29
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	12
21	22	23	24	25	26	27	28	29	30	16	17	18	19	20	10	9	8	7	6	5	4	3	2	1	15	14	13	12	11
22	23	24	25	26	27	28	29	8*	16	17	18	19	20	21	9	5*	7	6	2*	4	3	1*	13*	15	14	30*	12	11	10

LI(5,6) in S2 having pattern P2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	16
3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	18	19	20	21	22	23	24	25	26	27	28	29	30	16	17
4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	19	20	21	22	23	24	25	26	27	28	29	30	16	17	18
5	6	7	8	9	30*11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	10*16	17	18	19		
6	7	8	9	10	11	12	13	14	15	1	2	3	4	5	21	22	23	24	25	26	27	28	29	30	16	17	18	19	20
7	8	9	10	11	12	13	14	15	1	2	3	4	5	6	22	23	24	25	26	27	28	29	30	16	17	18	19	20	21
8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	23	24	25	26	27	28	29	30	16	17	18	19	20	21	22
9	10	11	12	13	14	15	1	2	3	4	5	6	7	8	24	25	26	27	28	29	30	16	17	18	19	20	21	22	23
10	11	12	13	14	15	1	2	3	4	5	6	7	8	9	25	26	27	28	29	30	16	17	18	19	20	21	22	23	24
11	12	13	14	15	1	2	3	4	5	6	7	8	9	10	26	27	28	29	30	16	17	18	19	20	21	22	23	24	25
12	13	14	15	1	2	3	4	5	6	7	8	9	10	11	27	28	29	30	16	17	18	19	20	21	22	23	24	25	26
13	14	15	1	2	3	4	5	6	7	8	9	10	11	12	28	29	30	16	17	18	19	20	21	22	23	24	25	26	27
14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	29	30	16	17	18	19	20	21	22	23	24	25	26	27	28
15	1	2	3	4	5	6	7	16*	9	10	11	12	13	14	30	8*17	18	19	20	21	22	23	24	25	26	27	28	29	
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	1





18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	12
21	22	23	24	25	26	27	28	29	30	16	17	18	19	20	10	9	8	7	6	5	4	3	2	1	15	14	13	12	11
22	23	24	25	26	27	28	29	8*16	17	18	19	20	21	9	6*	7	4*	5	3*	1*	2	14*15	30*13	12	11	10			
23	24	25	26	27	28	29	16*30*17	18	19	20	21	22	8	7	6	5	4	2*	3*	1	15	14	13	12	11	10	9		
24	25	26	27	28	29	30	12*17	18	19	20	21	22	23	7	16*	5	6*	3	4*	2*15	1*13	14*11	10	9	8				
25	26	27	28	29	30	16	17	18	19	20	21	22	23	24	6	5	4	3	2	1	15	14	13	12	11	10	9	8	7
26	27	28	29	30	16	17	18	19	20	21	22	23	24	25	5	4	3	2	1	15	14	13	12	11	10	9	8	7	6
27	28	29	30	16	17	18	19	20	21	22	23	24	25	26	4	3	2	1	15	14	13	12	11	10	9	8	7	6	5
28	29	30	16	17	18	19	20	21	22	23	24	25	26	27	3	2	1	15	14	13	12	11	10	9	8	7	6	5	4
29	30	16	17	18	19	20	21	22	23	24	25	26	27	28	2	1	15	14	13	12	11	10	9	8	7	6	5	4	3
30	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	15	14	13	12	11	10	9	8	7	6	5	4	3	2

LI(6,7) in S2 having pattern P7

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	16
3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	18	19	20	21	22	23	24	25	26	27	28	29	30	16	17
4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	19	20	21	22	23	24	25	26	27	28	29	30	16	17	18
5	6	7	8	9	10	11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	30	16	17	18	19
6	7	8	9	10	11	30*13	14	15	1	2	3	4	5	21	22	23	24	25	26	27	28	29	12*16	17	18	19	20		
7	8	9	10	11	12	13	14	15	1	2	3	4	5	6	22	23	24	25	26	27	28	29	30	16	17	18	19	20	21
8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	23	24	25	26	27	28	29	30	16	17	18	19	20	21	22
9	10	11	12	13	14	15	1	2	3	4	5	6	7	8	24	25	26	27	28	29	30	16	17	18	19	20	21	22	23
10	11	12	13	14	15	1	2	3	4	5	6	7	8	9	25	26	27	28	29	30	16	17	18	19	20	21	22	23	24
11	12	13	14	15	1	2	3	4	5	6	7	8	9	10	26	27	28	29	30	16	17	18	19	20	21	22	23	24	25
12	13	14	15	1	2	3	4	5	6	7	8	9	10	11	27	28	29	30	16	17	18	19	20	21	22	23	24	25	26
13	14	15	1	2	3	4	5	6	7	8	9	10	11	12	28	29	30	16	17	18	19	20	21	22	23	24	25	26	27
14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	29	30	16	17	18	19	20	21	22	23	24	25	26	27	28
15	1	2	3	4	5	6	7	16*	9	10	11	12	13	14	30	8*17	18	19	20	21	22	23	24	25	26	27	28	29	
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	12
21	22	23	24	25	26	27	28	29	30	16	17	18	19	20	10	9	8	7	6	5	4	3	2	1	15	14	13	12	11
22	23	24	25	26	27	28	29	8*16	17	18	19	20	21	9	5*	7	6	3*	4	15*	2	1	30*14	13	12	11	10		
23	24	25	26	27	28	29	16*30*17	18	19	20	21	22	8	7	6	5	4	2*	3*	1	15	14	13	12	11	10	9		
24	25	26	27	28	29	16*30*17	18	19	20	21	22	23	7	6	5	4	2*	3*	1	15	14	13	12	11	10	9	8		
25	26	27	28	29	30	12*17	18	19	20	21	22	23	24	6	16*	4	3	5*	1	2*14	13	15*11	10	9	8	7			
26	27	28	29	30	16	17	18	19	20	21	22	23	24	25	5	4	3	2	1	15	14	13	12	11	10	9	8	7	6
27	28	29	30	16	17	18	19	20	21	22	23	24	25	26	4	3	2	1	15	14	13	12	11	10	9	8	7	6	5
28	29	30	16	17	18	19	20	21	22	23	24	25	26	27	3	2	1	15	14	13	12	11	10	9	8	7	6	5	4
29	30	16	17	18	19	20	21	22	23	24	25	26	27	28	2	1	15	14	13	12	11	10	9	8	7	6	5	4	3
30	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	15	14	13	12	11	10	9	8	7	6	5	4	3	2

LI(6,8) in S2 having pattern P1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	16
3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	18	19	20	21	22	23	24	25	26	27	28	29	30	16	17
4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	19	20	21	22	23	24	25	26	27	28	29	30	16	17	18
5	6	7	8	9	10	11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	30	16	17	18	19
6	7	8	9	10	11	12	30*14	15	1	2	3	4	5	21	22	23	24	25	26	27	28	29	13*16	17	18	19	20		
7	8	9	10	11	12	13	14	15	1	2	3	4	5	6	22	23	24	25	26	27	28	29	30	16	17	18	19	20	21
8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	23	24	25	26	27	28	29	30	16	17	18	19	20	21	22
9	10	11	12	13	14	15	1	2	3	4	5	6	7	8	24	25	26	27	28	29	30	16	17	18	19	20	21	22	23
10	11	12	13	14	15	1	2	3	4	5	6	7	8	9	25	26	27	28	29	30	16	17	18	19	20	21	22	23	24
11	12	13	14	15	1	2	3	4	5	6	7	8	9	10	26	27	28	29	30	16	17	18	19	20	21	22	23	24	25

12	13	14	15	1	2	3	4	5	6	7	8	9	10	11	27	28	29	30	16	17	18	19	20	21	22	23	24	25	26
13	14	15	1	2	3	4	5	6	7	8	9	10	11	12	28	29	30	16	17	18	19	20	21	22	23	24	25	26	27
14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	29	30	16	17	18	19	20	21	22	23	24	25	26	27	28
15	1	2	3	4	5	6	7	16*	9	10	11	12	13	14	30	8*17	18	19	20	21	22	23	24	25	26	27	28	29	
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	12
21	22	23	24	25	26	27	28	29	30	16	17	18	19	20	10	9	8	7	6	5	4	3	2	1	15	14	13	12	11
22	23	24	25	26	27	28	29	8*16	17	18	19	20	21	9	6*	7	4*	5	2*	3	15*	1	30*14	13	12	11	10		
23	24	25	26	27	28	29	16*30*17	18	19	20	21	22	8	7	6	5	4	3	2	1	15	14	13	12	11	10	9		
24	25	26	27	28	29	30	13*17	18	19	20	21	22	23	7	16*	5	6*	3	4*	1	2*14	15*12	11	10	9	8			
25	26	27	28	29	30	16	17	18	19	20	21	22	23	24	6	5	4	3	2	1	15	14	13	12	11	10	9	8	7
26	27	28	29	30	16	17	18	19	20	21	22	23	24	25	5	4	3	2	1	15	14	13	12	11	10	9	8	7	6
27	28	29	30	16	17	18	19	20	21	22	23	24	25	26	4	3	2	1	15	14	13	12	11	10	9	8	7	6	5
28	29	30	16	17	18	19	20	21	22	23	24	25	26	27	3	2	1	15	14	13	12	11	10	9	8	7	6	5	4
29	30	16	17	18	19	20	21	22	23	24	25	26	27	28	2	1	15	14	13	12	11	10	9	8	7	6	5	4	3
30	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	15	14	13	12	11	10	9	8	7	6	5	4	3	2

LI(7,7) in S2 having pattern P2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	16
3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	18	19	20	21	22	23	24	25	26	27	28	29	30	16	17
4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	19	20	21	22	23	24	25	26	27	28	29	30	16	17	18
5	6	7	8	9	10	11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	30	16	17	18	19
6	7	8	9	10	11	12	13	14	15	1	2	3	4	5	21	22	23	24	25	26	27	28	29	30	16	17	18	19	20
7	8	9	10	11	12	30*14	15	1	2	3	4	5	6	22	23	24	25	26	27	28	29	13*16	17	18	19	20	21		
8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	23	24	25	26	27	28	29	30	16	17	18	19	20	21	22
9	10	11	12	13	14	15	1	2	3	4	5	6	7	8	24	25	26	27	28	29	30	16	17	18	19	20	21	22	23
10	11	12	13	14	15	1	2	3	4	5	6	7	8	9	25	26	27	28	29	30	16	17	18	19	20	21	22	23	24
11	12	13	14	15	1	2	3	4	5	6	7	8	9	10	26	27	28	29	30	16	17	18	19	20	21	22	23	24	25
12	13	14	15	1	2	3	4	5	6	7	8	9	10	11	27	28	29	30	16	17	18	19	20	21	22	23	24	25	26
13	14	15	1	2	3	4	5	6	7	8	9	10	11	12	28	29	30	16	17	18	19	20	21	22	23	24	25	26	27
14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	29	30	16	17	18	19	20	21	22	23	24	25	26	27	28
15	1	2	3	4	5	6	7	16*	9	10	11	12	13	14	30	8*17	18	19	20	21	22	23	24	25	26	27	28	29	
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	12
21	22	23	24	25	26	27	28	29	30	16	17	18	19	20	10	9	8	7	6	5	4	3	2	1	15	14	13	12	11
22	23	24	25	26	27	28	29	8*16	17	18	19	20	21	9	5*	7	6	4*	1*	3	2	30*15	14	13	12	11	10		
23	24	25	26	27	28	29	16*30*17	18	19	20	21	22	8	7	6	5	3*	4*	2	1	15	14	13	12	11	10	9		
24	25	26	27	28	29	16*30*17	18	19	20	21	22	23	7	6	5	4	2*	3*	1	15	14	13	12	11	10	9	8		
25	26	27	28	29	30	13*17	18	19	20	21	22	23	24	6	16*	4	3	5*	2*15	14	1*12	11	10	9	8	7			
26	27	28	29	30	16	17	18	19	20	21	22	23	24	25	5	4	3	2	1	15	14	13	12	11	10	9	8	7	6
27	28	29	30	16	17	18	19	20	21	22	23	24	25	26	4	3	2	1	15	14	13	12	11	10	9	8	7	6	5
28	29	30	16	17	18	19	20	21	22	23	24	25	26	27	3	2	1	15	14	13	12	11	10	9	8	7	6	5	4
29	30	16	17	18	19	20	21	22	23	24	25	26	27	28	2	1	15	14	13	12	11	10	9	8	7	6	5	4	3
30	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	15	14	13	12	11	10	9	8	7	6	5	4	3	2

LI(7,8) in S2 having pattern P3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	16
3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	18	19	20	21	22	23	24	25	26	27	28	29	30	16	17
4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	19	20	21	22	23	24	25	26	27	28	29	30	16	17	18
5	6	7	8	9	10	11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	30	16	17	18	19















3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	18	19	20	21	22	23	24	25	26	27	28	29	30	16	17
4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	19	20	21	22	23	24	25	26	27	28	29	30	16	17	18
5	6	7	8	9	10	11	12	13	14	15	1	2	3	4	20	21	22	23	24	25	26	27	28	29	30	16	17	18	19
6	7	8	9	10	11	12	13	14	15	1	2	3	4	5	21	22	23	24	25	26	27	28	29	30	16	17	18	19	20
7	8	9	10	11	12	13	14	15	1	2	3	4	5	6	22	23	24	25	26	27	28	29	30	16	17	18	19	20	21
8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	23	24	25	26	27	28	29	30	16	17	18	19	20	21	22
9	10	11	12	13	14	15	1	2	3	4	5	6	7	8	24	25	26	27	28	29	30	16	17	18	19	20	21	22	23
10	11	12	13	14	15	1	2	3	4	5	6	7	8	9	25	26	27	28	29	30	16	17	18	19	20	21	22	23	24
11	12	13	14	15	1	2	3	4	5	6	7	8	9	22*	26	27	28	29	30	16	17	18	19	20	21	10*	23	24	25
12	13	14	15	1	2	3	4	5	6	7	8	9	10	11	27	28	29	30	16	17	18	19	20	21	22	23	24	25	26
13	14	15	1	2	3	4	5	6	7	8	9	10	11	12	28	29	30	16	17	18	19	20	21	22	23	24	25	26	27
14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	29	30	16	17	18	19	20	21	22	23	24	25	26	27	28
15	1	2	3	4	5	6	7	8	9	22*	11	12	13	10*	30	16	17	18	19	20	21	14*	23	24	25	26	27	28	29
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
18	19	20	21	22	23	24	25	26	27	28	29	30	16	17	13	12	11	10	9	8	7	6	5	4	3	2	1	15	14
19	20	21	22	23	24	25	26	27	28	29	30	16	17	18	12	11	10	9	8	7	6	5	4	3	2	1	15	14	13
20	21	22	23	24	25	26	27	28	29	30	16	17	18	19	11	10	9	8	7	6	5	4	3	2	1	15	14	13	12
21	22	23	24	25	26	27	28	29	30	16	17	18	19	20	10	9	8	7	6	5	4	3	2	1	15	14	13	12	11
22	23	24	25	26	27	28	29	30	16	17	18	19	20	21	9	8	7	6	5	4	3	2	1	15	14	13	12	11	10
23	24	25	26	27	28	29	30	16	17	18	19	20	21	14*	8	7	6	5	4	3	2	1	15	13*	12*	22*	11	10	9
24	25	26	27	28	29	30	16	17	18	19	20	21	22	23	7	6	5	4	3	2	1	15	11*	14*	13*	12*	10	9	8
25	26	27	28	29	30	16	17	18	19	20	21	22	23	24	6	5	4	3	2	1	15	13*	14*	12*	10*	11*	9	8	7
26	27	28	29	30	16	17	18	19	20	21	22	23	24	25	5	4	3	2	1	15	14	12*	13*	10*	11*	9	8	7	6
27	28	29	30	16	17	18	19	20	21	10*	23	24	25	26	4	3	2	1	15	14	13	22*	12*	11*	9	8	7	6	5
28	29	30	16	17	18	19	20	21	22	23	24	25	26	27	3	2	1	15	14	13	12	11	10	9	8	7	6	5	4
29	30	16	17	18	19	20	21	22	23	24	25	26	27	28	2	1	15	14	13	12	11	10	9	8	7	6	5	4	3
30	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	15	14	13	12	11	10	9	8	7	6	5	4	3	2

# Latin Interchanges for MBC(2,17)

.....

MBC(2,17)

```

1 2 3 4 5 6 7 8 910111213141516171819202122232425262728293031323334
2 3 4 5 6 7 8 91011121314151617 11920212223242526272829303132333418
3 4 5 6 7 8 91011121314151617 1 22021222324252627282930313233341819
4 5 6 7 8 91011121314151617 1 2 32122232425262728293031323334181920
5 6 7 8 91011121314151617 1 2 3 42223242526272829303132333418192021
6 7 8 91011121314151617 1 2 3 4 52324252627282930313233341819202122
7 8 91011121314151617 1 2 3 4 5 62425262728293031323334181920212223
8 91011121314151617 1 2 3 4 5 6 72526272829303132333418192021222324
91011121314151617 1 2 3 4 5 6 7 82627282930313233341819202122232425
1011121314151617 1 2 3 4 5 6 7 8 92728293031323334181920212223242526
11121314151617 1 2 3 4 5 6 7 8 9102829303132333418192021222324252627
121314151617 1 2 3 4 5 6 7 8 910112930313233341819202122232425262728
1314151617 1 2 3 4 5 6 7 8 91011123031323334181920212223242526272829
14151617 1 2 3 4 5 6 7 8 9101112133132333418192021222324252627282930
151617 1 2 3 4 5 6 7 8 910111213143233341819202122232425262728293031
1617 1 2 3 4 5 6 7 8 91011121314153334181920212223242526272829303132
17 12 3 4 5 6 7 8 9101112131415163418192021222324252627282930313233
18192021222324252627282930313233341716151413121110 9 8 7 6 5 4 3 2 1
192021222324252627282930313233341816151413121110 9 8 7 6 5 4 3 2 117
2021222324252627282930313233341819151413121110 9 8 7 6 5 4 3 2 11716
21222324252627282930313233341819201413121110 9 8 7 6 5 4 3 2 1171615
222324252627282930313233341819202113121110 9 8 7 6 5 4 3 2 117161514
2324252627282930313233341819202122121110 9 8 7 6 5 4 3 2 11716151413
24252627282930313233341819202122231110 9 8 7 6 5 4 3 2 1171615141312
252627282930313233341819202122232410 9 8 7 6 5 4 3 2 117161514131211
2627282930313233341819202122232425 9 8 7 6 5 4 3 2 11716151413121110
2728293031323334181920212223242526 8 7 6 5 4 3 2 11716151413121110 9
2829303132333418192021222324252627 7 6 5 4 3 2 11716151413121110 9 8
2930313233341819202122232425262728 6 5 4 3 2 11716151413121110 9 8 7
3031323334181920212223242526272829 5 4 3 2 11716151413121110 9 8 7 6
3132333418192021222324252627282930 4 3 2 11716151413121110 9 8 7 6 5
3233341819202122232425262728293031 3 2 11716151413121110 9 8 7 6 5 4
3334181920212223242526272829303132 2 11716151413121110 9 8 7 6 5 4 3
3418192021222324252627282930313233 11716151413121110 9 8 7 6 5 4 3 2

```

LI(5,7) in S2 having pattern P7

```

1 2 3 4 5 6 7 8 910111213141516171819202122232425262728293031323334
2 3 4 5 6 7 8 91011121314151617 11920212223242526272829303132333418
3 4 5 6 7 8 91011121314151617 1 22021222324252627282930313233341819
4 5 6 7 8 91011121314151617 1 2 32122232425262728293031323334181920
5 6 7 8 91034121314151617 1 2 3 42223242526272829303132331118192021
6 7 8 91011121314151617 1 2 3 4 52324252627282930313233341819202122
7 8 91011121314151617 1 2 3 4 5 62425262728293031323334181920212223
8 91011121314151617 1 2 3 4 5 6 72526272829303132333418192021222324
91011121314151617 1 2 3 4 5 6 7 82627282930313233341819202122232425
1011121314151617 1 2 3 4 5 6 7 8 92728293031323334181920212223242526
11121314151617 1 2 3 4 5 6 7 8 9102829303132333418192021222324252627
121314151617 1 2 3 4 5 6 7 8 910112930313233341819202122232425262728
1314151617 1 2 3 4 5 6 7 8 91011123031323334181920212223242526272829
14151617 1 2 3 4 5 6 7 8 9101112133132333418192021222324252627282930
151617 1 2 3 4 5 6 7 8 910111213143233341819202122232425262728293031
1617 1 2 3 4 5 6 7 8 91011121314153334181920212223242526272829303132
17 1 2 3 4 5 6 7 8181011121314151634 9192021222324252627282930313233
18192021222324252627282930313233341716151413121110 9 8 7 6 5 4 3 2 1

```

192021222324252627282930313233341816151413121110 9 8 7 6 5 4 3 2 117  
2021222324252627282930313233341819151413121110 9 8 7 6 5 4 3 2 11716  
21222324252627282930313233341819201413121110 9 8 7 6 5 4 3 2 1171615  
222324252627282930313233341819202113121110 9 8 7 6 5 4 3 2 117161514  
2324252627282930313233341819202122121110 9 8 7 6 5 4 3 2 11716151413  
24252627282930313233341819202122231110 9 8 7 6 5 4 3 2 1171615141312  
252627282930313233 91819202122232410 5 8 7 6 2 4 315 117163414131211  
2627282930313233183419202122232425 9 8 7 6 5 4 3 1 21716151413121110  
2728293031323318341920212223242526 8 7 6 5 4 3 1 21716151413121110 9  
2829303132331834192021222324252627 7 6 5 4 3 1 21716151413121110 9 8  
2930313233341119202122232425262728 618 4 3 2 51716 11413121510 9 8 7  
3031323334181920212223242526272829 5 4 3 2 11716151413121110 9 8 7 6  
3132333418192021222324252627282930 4 3 2 11716151413121110 9 8 7 6 5  
3233341819202122232425262728293031 3 2 11716151413121110 9 8 7 6 5 4  
3334181920212223242526272829303132 2 11716151413121110 9 8 7 6 5 4 3  
3418192021222324252627282930313233 11716151413121110 9 8 7 6 5 4 3 2

LI(17,17) in S4 having pattern P2

1 2 3 4 5 6 7 8 910111213141516171819202122232425262728293031323334  
2 3 4 5 6 7 8 91011121314151617 11920212223242526272829303132333418  
3 45 6 7 8 91011121314151617 1 22021222324252627282930313233341819  
4 5 6 7 8 91011121314151617 1 2 32122232425262728293031323334181920  
5 6 7 8 91011121314151617 1 2 3 42223242526272829303132333418192021  
6 7 8 91011121314151617 1 2 3 4 52324252627282930313233341819202122  
7 8 91011121314151617 1 2 3 4 5 62425262728293031323334181920212223  
8 91011121314151617 1 2 3 4 5 6 72526272829303132333418192021222324  
91011121314151617 1 2 3 4 5 6 7 82627282930313233341819202122232425  
1011121314151617 1 2 3 4 5 6 7 8 92728293031323334181920212223242526  
11121314151617 1 2 3 4 5 6 7 8 9102829303132333418192021222324252627  
121314151617 1 2 3 4 5 6 7 8 910252930313233341819202122232411262728  
1314151617 1 2 3 4 5 6 7 8 91011123031323334181920212223242526272829  
14151617 1 2 3 4 5 6 7 8 9101112133132333418192021222324252627282930  
151617 1 2 3 4 5 6 7 8 910111213143233341819202122232425262728293031  
1617 1 2 3 4 5 6 7 8 91011121314153334181920212223242526272829303132  
17 1 2 3 4 5 6 7 8 9102512131415113418192021222324162627282930313233  
18192021222324252627282930313233341716151413121110 9 8 7 6 5 4 3 2 1  
192021222324252627282930313233341816151413121110 9 8 7 6 5 4 3 2 117  
2021222324252627282930313233341819151413121110 9 8 7 6 5 4 3 2 11716  
21222324252627282930313233341819201413121110 9 8 7 6 5 4 3 2 1171615  
222324252627282930313233341819202113121110 9 8 7 6 5 4 3 2 117161514  
2324252627282930313233341819202122121110 9 8 7 6 5 4 3 2 11716151413  
24252627282930313233341819202122231110 9 8 7 6 5 4 3 2 1171615141312  
252627282930313233341819202122232410 9 8 7 6 5 4 3 2 117161514131211  
2627282930313233341819202122232416 9 8 7 6 5 4 3 2 11715141325121110  
2728293031323334181920212223242526 8 7 6 5 4 3 2 11712161514131110 9  
2829303132333418192021222324252627 7 6 5 4 3 2 11715161413111210 9 8  
2930313233341819202122232425262728 6 5 4 3 2 11716141513111210 9 8 7  
3031323334181920212223242526272829 5 4 3 2 11716151314111210 9 8 7 6  
3132333418192021222324112627282930 4 3 2 11716151425131210 9 8 7 6 5  
3233341819202122232425262728293031 3 2 11716151413121110 9 8 7 6 5 4  
3334181920212223242526272829303132 2 11716151413121110 9 8 7 6 5 4 3  
3418192021222324252627282930313233 11716151413121110 9 8 7 6 5 4 3 2

Latin Interchanges for MBC(2,19)

.....

MBC(2,19)

1 2 3 4 5 6 7 8 910111213141516171819	20212223242526272829303132333435363738
2 3 4 5 6 7 8 910111213141516171819 1	21222324252627282930313233343536373820
3 4 5 6 7 8 910111213141516171819 1 2	22232425262728293031323334353637382021
4 5 6 7 8 910111213141516171819 1 2 3	23242526272829303132333435363738202122
5 6 7 8 910111213141516171819 1 2 3 4	24252627282930313233343536373820212223
6 7 8 910111213141516171819 1 2 3 4 5	25262728293031323334353637382021222324
7 8 910111213141516171819 1 2 3 4 5 6	26272829303132333435363738202122232425
8 910111213141516171819 1 2 3 4 5 6 7	27282930313233343536373820212223242526
910111213141516171819 1 2 3 4 5 6 7 8	28293031323334353637382021222324252627
10111213141516171819 1 2 3 4 5 6 7 8 9	29303132333435363738202122232425262728
111213141516171819 1 2 3 4 5 6 7 8 910	30313233343536373820212223242526272829
1213141516171819 1 2 3 4 5 6 7 8 91011	31323334353637382021222324252627282930
13141516171819 1 2 3 4 5 6 7 8 9101112	32333435363738202122232425262728293031
141516171819 1 2 3 4 5 6 7 8 910111213	33343536373820212223242526272829303132
1516171819 1 2 3 4 5 6 7 8 91011121314	34353637382021222324252627282930313233
16171819 1 2 3 4 5 6 7 8 9101112131415	35363738202122232425262728293031323334
171819 1 2 3 4 5 6 7 8 910111213141516	36373820212223242526272829303132333435
1819 1 2 3 4 5 6 7 8 91011121314151617	37382021222324252627282930313233343536
19 1 2 3 4 5 6 7 8 9101112131415161718	38202122232425262728293031323334353637

20212223242526272829303132333435363738	19181716151413121110 9 8 7 6 5 4 3 2 1
21222324252627282930313233343536373820	181716151413121110 9 8 7 6 5 4 3 2 119
22232425262728293031323334353637382021	1716151413121110 9 8 7 6 5 4 3 2 11918
23242526272829303132333435363738202122	16151413121110 9 8 7 6 5 4 3 2 1191817
24252627282930313233343536373820212223	151413121110 9 8 7 6 5 4 3 2 119181716
25262728293031323334353637382021222324	1413121110 9 8 7 6 5 4 3 2 11918171615
26272829303132333435363738202122232425	13121110 9 8 7 6 5 4 3 2 1191817161514
27282930313233343536373820212223242526	121110 9 8 7 6 5 4 3 2 119181716151413
28293031323334353637382021222324252627	1110 9 8 7 6 5 4 3 2 11918171615141312
29303132333435363738202122232425262728	10 9 8 7 6 5 4 3 2 1191817161514131211
30313233343536373820212223242526272829	9 8 7 6 5 4 3 2 119181716151413121110
31323334353637382021222324252627282930	8 7 6 5 4 3 2 119181716151413121110 9
32333435363738202122232425262728293031	7 6 5 4 3 2 119181716151413121110 9 8
33343536373820212223242526272829303132	6 5 4 3 2 119181716151413121110 9 8 7
34353637382021222324252627282930313233	5 4 3 2 119181716151413121110 9 8 7 6
35363738202122232425262728293031323334	4 3 2 119181716151413121110 9 8 7 6 5
36373820212223242526272829303132333435	3 2 119181716151413121110 9 8 7 6 5 4
37382021222324252627282930313233343536	2 119181716151413121110 9 8 7 6 5 4 3
38202122232425262728293031323334353637	119181716151413121110 9 8 7 6 5 4 3 2

LI(4,8) in S2 having pattern P6

1 2 3 4 5 6 7 8 910111213141516171819	20212223242526272829303132333435363738
2 3 4 5 6 7 8 910111213141516171819 1	21222324252627282930313233343536373820
3 4 5 6 7 8 910111213141516171819 1 2	22232425262728293031323334353637382021
4 5 6 7 8 910381213141516171819 1 2 3	23242526272829303132333435363711202122
5 6 7 8 910111213141516171819 1 2 3 4	24252627282930313233343536373820212223
6 7 8 910111213141516171819 1 2 3 4 5	25262728293031323334353637382021222324
7 8 910111213141516171819 1 2 3 4 5 6	26272829303132333435363738202122232425
8 910111213141516171819 1 2 3 4 5 6 7	27282930313233343536373820212223242526
910111213141516171819 1 2 3 4 5 6 7 8	28293031323334353637382021222324252627
10111213141516171819 1 2 3 4 5 6 7 8 9	29303132333435363738202122232425262728
111213141516171819 1 2 3 4 5 6 7 8 910	30313233343536373820212223242526272829
1213141516171819 1 2 3 4 5 6 7 8 91011	31323334353637382021222324252627282930
13141516171819 1 2 3 4 5 6 7 8 9101112	32333435363738202122232425262728293031



141516171819 1 2 3 4 5 6 7 8 910111213 33343536373820212223242526272829303132  
1516171819 1 2 3 4 5 6 7 8 91011121314 34353637382021222324252627282930313233  
16171819 1 2 3 4 5 6 7 8 9101112131415 35363738202122232425262728293031323334  
171819 1 2 3 4 5 6 7 8 910111213141516 36373820212223242526272829303132333435  
1819 1 2 3 4 5 6 7 8 91011121314151617 37382021222324252627282930313233343536  
19 1 2 3 4 5 6 7 8 9201112131415161718 38102122232425262728293031323334353637

20212223242526272829303132333435363738 19181716151413121110 9 8 7 6 5 4 3 2 1  
21222324252627282930313233343536373820 181716151413121110 9 8 7 6 5 4 3 2 119  
22232425262728293031323334353637382021 1716151413121110 9 8 7 6 5 4 3 2 11918  
23242526272829303132333435363738202122 16151413121110 9 8 7 6 5 4 3 2 1191817  
24252627282930313233343536373820212223 151413121110 9 8 7 6 5 4 3 2 119181716  
25262728293031323334353637382021222324 1413121110 9 8 7 6 5 4 3 2 11918171615  
26272829303132333435363738202122232425 13121110 9 8 7 6 5 4 3 2 1191817161514  
27282930313233343536373820212223242526 121110 9 8 7 6 5 4 3 2 119181716151413  
28293031323334353637102021222324252627 11 6 9 8 7 2 5 4 319 11518171638141312  
29303132333435363720382122232425262728 10 9 8 7 6 5 4 3 2 1191817161514131211  
30313233343536372038212223242526272829 9 8 7 6 5 4 319 1 2181716151413121110  
31323334353637203821222324252627282930 8 7 6 5 4 3 2 119181716151413121110 9  
32333435363738112122232425262728293031 720 5 4 3 6 1 2181716191413121510 9 8  
33343536373820212223242526272829303132 6 5 4 3 2 119181716151413121110 9 8 7  
34353637382021222324252627282930313233 5 4 3 2 119181716151413121110 9 8 7 6  
35363738202122232425262728293031323334 4 3 2 119181716151413121110 9 8 7 6 5  
36373820212223242526272829303132333435 3 2 119181716151413121110 9 8 7 6 5 4  
37382021222324252627282930313233343536 2 119181716151413121110 9 8 7 6 5 4 3  
38202122232425262728293031323334353637 119181716151413121110 9 8 7 6 5 4 3 2

LI(7,8) in S2 having pattern P7

1 2 3 4 5 6 7 8 910111213141516171819 20212223242526272829303132333435363738  
2 3 4 5 6 7 8 910111213141516171819 1 21222324252627282930313233343536373820  
3 4 5 6 7 8 910111213141516171819 1 2 22232425262728293031323334353637382021  
4 5 6 7 8 910111213141516171819 1 2 3 23242526272829303132333435363738202122  
5 6 7 8 910111213141516171819 1 2 3 4 24252627282930313233343536373820212223  
6 7 8 910111213141516171819 1 2 3 4 5 25262728293031323334353637382021222324  
7 8 910111213381516171819 1 2 3 4 5 6 26272829303132333435363714202122232425  
8 910111213141516171819 1 2 3 4 5 6 7 27282930313233343536373820212223242526  
910111213141516171819 1 2 3 4 5 6 7 8 28293031323334353637382021222324252627  
10111213141516171819 1 2 3 4 5 6 7 8 9 29303132333435363738202122232425262728  
111213141516171819 1 2 3 4 5 6 7 8 910 30313233343536373820212223242526272829  
1213141516171819 1 2 3 4 5 6 7 8 91011 31323334353637382021222324252627282930  
13141516171819 1 2 3 4 5 6 7 8 9101112 32333435363738202122232425262728293031  
141516171819 1 2 3 4 5 6 7 8 910111213 33343536373820212223242526272829303132  
1516171819 1 2 3 4 5 6 7 8 91011121314 34353637382021222324252627282930313233  
16171819 1 2 3 4 5 6 7 8 9101112131415 35363738202122232425262728293031323334  
171819 1 2 3 4 5 6 7 8 910111213141516 36373820212223242526272829303132333435  
1819 1 2 3 4 5 6 7 8 91011121314151617 37382021222324252627282930313233343536  
19 1 2 3 4 5 6 7 8 9201112131415161718 38102122232425262728293031323334353637

20212223242526272829303132333435363738 19181716151413121110 9 8 7 6 5 4 3 2 1  
21222324252627282930313233343536373820 181716151413121110 9 8 7 6 5 4 3 2 119  
22232425262728293031323334353637382021 1716151413121110 9 8 7 6 5 4 3 2 11918  
23242526272829303132333435363738202122 16151413121110 9 8 7 6 5 4 3 2 1191817  
24252627282930313233343536373820212223 151413121110 9 8 7 6 5 4 3 2 119181716  
25262728293031323334353637382021222324 1413121110 9 8 7 6 5 4 3 2 11918171615  
26272829303132333435363738202122232425 13121110 9 8 7 6 5 4 3 2 1191817161514  
27282930313233343536373820212223242526 121110 9 8 7 6 5 4 3 2 119181716151413  
28293031323334353637102021222324252627 11 6 9 8 7 3 5 418 2 11938171615141312  
29303132333435363720382122232425262728 10 9 8 7 6 5 4 2 3 1191817161514131211  
30313233343536372038212223242526272829 9 8 7 6 5 4 2 3 119181716151413121110

31323334353637203821222324252627282930	8 7 6 5 4 2 3 119181716151413121110 9
32333435363738142122232425262728293031	720 5 4 3 6 119 21716151813121110 9 8
33343536373820212223242526272829303132	6 5 4 3 2 119181716151413121110 9 8 7
34353637382021222324252627282930313233	5 4 3 2 119181716151413121110 9 8 7 6
35363738202122232425262728293031323334	4 3 2 119181716151413121110 9 8 7 6 5
36373820212223242526272829303132333435	3 2 119181716151413121110 9 8 7 6 5 4
37382021222324252627282930313233343536	2 119181716151413121110 9 8 7 6 5 4 3
38202122232425262728293031323334353637	119181716151413121110 9 8 7 6 5 4 3 2

## MBC (2, 21)

222324252627282930313233343536373839404142	212019181716151413121110 9 8 7 6 5 4 3 2 1
232425262728293031323334353637383940414222	2019181716151413121110 9 8 7 6 5 4 3 2 121
242526272829303132333435363738394041422223	19181716151413121110 9 8 7 6 5 4 3 2 12120
252627282930313233343536373839404142222324	181716151413121110 9 8 7 6 5 4 3 2 1212019
262728293031323334353637383940414222232425	1716151413121110 9 8 7 6 5 4 3 2 121201918
272829303132333435363738394041422223242526	16151413121110 9 8 7 6 5 4 3 2 12120191817
282930313233343536373839404142222324252627	151413121110 9 8 7 6 5 4 3 2 1212019181716
293031323334353637383940414222232425262728	1413121110 9 8 7 6 5 4 3 2 121201918171615
303132333435363738394041422223242526272829	13121110 9 8 7 6 5 4 3 2 12120191817161514
313233343536373839404142222324252627282930	121110 9 8 7 6 5 4 3 2 1212019181716151413
323334353637383940414222232425262728293031	1110 9 8 7 6 5 4 3 2 121201918171615141312
333435363738394041422223242526272829303132	10 9 8 7 6 5 4 3 2 12120191817161514131211
343536373839404142222324252627282930313233	9 8 7 6 5 4 3 2 1212019181716151413121110
353637383940414222232425262728293031323334	8 7 6 5 4 3 2 1212019181716151413121110 9
363738394041422223242526272829303132333435	7 6 5 4 3 2 1212019181716151413121110 9 8
373839404142222324252627282930313233343536	6 5 4 3 2 1212019181716151413121110 9 8 7
383940414222232425262728293031323334353637	5 4 3 2 1212019181716151413121110 9 8 7 6
394041422223242526272829303132333435363738	4 3 2 1212019181716151413121110 9 8 7 6 5
404142222324252627282930313233343536373839	3 2 1212019181716151413121110 9 8 7 6 5 4
414222232425262728293031323334353637383940	2 1212019181716151413121110 9 8 7 6 5 4 3
422223242526272829303132333435363738394041	1212019181716151413121110 9 8 7 6 5 4 3 2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	1	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	22
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	1	2	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	22	23
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	1	2	3	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	22	23	24
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	1	2	3	4	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	22	23	24	25
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	1	2	3	4	5	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	22	23	24	25	26
7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	1	2	3	4	5	6	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	22	23	24	25	26	27
8	9	10	11	12	13	14	15	16	17	18	19	20	21	1	2	3	4	5	6	7	29	30	31	32	33	34	35	36	37	38	39	40	41	42	22	23	24	25	26	27	28
9	10	11	12	13	14	15	16	17	18	19	20	21	1	2	3	4	5	6	7	8	30	31	32	33	34	35	36	37	38	39	40	41	42	22	23	24	25	26	27	28	29
10	11	12	13	14	15	16	17	18	19	20	21	1	2	3	4	5	6	7	8	9	31	32	33	34	35	36	37	38	39	40	41	42	22	23	24	25	26	27	28	29	30

1112131415161718192021 1 2 3 4 5 6 7 8 910 323334353637383940414222232425262728293031  
12131415161718192021 1 2 3 4 5 6 7 8 91011 333435363738394041422223242526272829303132  
131415161718192021 1 2 3 4 5 6 7 8 9101112 343536373839404142222324252627282930313233  
1415161718192021 1 2 3 4 5 6 7 8 910111213 353637383940414222232425262728293031323334  
15161718192021 1 2 3 4 5 6 7 8 91011121314 363738394041422223242526272829303132333435  
161718192021 1 2 3 4 5 6 7 8 9101112131415 373839404142222324252627282930313233343536  
1718192021 1 2 3 4 5 6 7 8 910111213141516 383940414222232425262728293031323334353637  
18192021 1 2 3 4 5 6 7 8 91011121314151617 394041422223242526272829303132333435363738  
192021 1 2 3 4 5 6 7 8 9101112131415161718 404142222324252627282930313233343536373839  
2021 1 2 3 4 5 6 7 8 910111213141516171819 414222232425262728293031323334353637383940  
21 1 2 3 4 5 6 7 8 91022121314151617181920 421123242526272829303132333435363738394041

222324252627282930313233343536373839404142 212019181716151413121110 9 8 7 6 5 4 3 2 1  
232425262728293031323334353637383940414222 2019181716151413121110 9 8 7 6 5 4 3 2 121  
242526272829303132333435363738394041422223 19181716151413121110 9 8 7 6 5 4 3 2 12120  
252627282930313233343536373839404142222324 181716151413121110 9 8 7 6 5 4 3 2 1212019  
262728293031323334353637383940414222232425 1716151413121110 9 8 7 6 5 4 3 2 121201918  
272829303132333435363738394041422223242526 16151413121110 9 8 7 6 5 4 3 2 12120191817  
282930313233343536373839404142222324252627 151413121110 9 8 7 6 5 4 3 2 1212019181716  
293031323334353637383940414222232425262728 1413121110 9 8 7 6 5 4 3 2 121201918171615  
303132333435363738394041422223242526272829 13121110 9 8 7 6 5 4 3 2 12120191817161514  
313233343536373839404111222324252627282930 12 510 9 8 7 6 3 418 2 1212019421716151413  
323334353637383940412242232425262728293031 1110 9 8 7 6 5 4 3 2 121201918171615141312  
333435363738394041224223242526272829303132 10 9 8 7 6 5 4 1 2 32120191817161514131211  
343536373839404122422324252627282930313233 9 8 7 6 5 4 3 2 1212019181716151413121110  
353637383940412242232425262728293031323334 8 7 6 5 4 3 2 2021 119181716151413121110 9  
363738394041224223242526272829303132333435 7 6 5 4 3 2 1212019181716151413121110 9 8  
373839404142122324252627282930313233343536 622 4 3 2 121 519201716151413181110 9 8 7  
383940414222232425262728293031323334353637 5 4 3 2 1212019181716151413121110 9 8 7 6  
394041422223242526272829303132333435363738 4 3 2 1212019181716151413121110 9 8 7 6 5  
404142222324252627282930313233343536373839 3 2 1212019181716151413121110 9 8 7 6 5 4  
414222232425262728293031323334353637383940 2 1212019181716151413121110 9 8 7 6 5 4 3  
422223242526272829303132333435363738394041 1212019181716151413121110 9 8 7 6 5 4 3 2

LI(6,8) in S2 having pattren P7

1 2 3 4 5 6 7 8 9101112131415161718192021 222324252627282930313233343536373839404142  
2 3 4 5 6 7 8 9101112131415161718192021 1 232425262728293031323334353637383940414222  
3 4 5 6 7 8 9101112131415161718192021 1 2 242526272829303132333435363738394041422223  
4 5 6 7 8 9101112131415161718192021 1 2 3 252627282930313233343536373839404142222324  
5 6 7 8 9101112131415161718192021 1 2 3 4 262728293031323334353637383940414222232425  
6 7 8 9101112421415161718192021 1 2 3 4 5 272829303132333435363738394041132223242526  
7 8 9101112131415161718192021 1 2 3 4 5 6 282930313233343536373839404142222324252627  
8 9101112131415161718192021 1 2 3 4 5 6 7 293031323334353637383940414222232425262728  
9101112131415161718192021 1 2 3 4 5 6 7 8 303132333435363738394041422223242526272829  
101112131415161718192021 1 2 3 4 5 6 7 8 9 313233343536373839404142222324252627282930  
1112131415161718192021 1 2 3 4 5 6 7 8 910 323334353637383940414222232425262728293031  
12131415161718192021 1 2 3 4 5 6 7 8 91011 333435363738394041422223242526272829303132  
131415161718192021 1 2 3 4 5 6 7 8 9101112 343536373839404142222324252627282930313233  
1415161718192021 1 2 3 4 5 6 7 8 910111213 353637383940414222232425262728293031323334  
15161718192021 1 2 3 4 5 6 7 8 91011121314 363738394041422223242526272829303132333435  
161718192021 1 2 3 4 5 6 7 8 9101112131415 373839404142222324252627282930313233343536  
1718192021 1 2 3 4 5 6 7 8 910111213141516 383940414222232425262728293031323334353637  
18192021 1 2 3 4 5 6 7 8 91011121314151617 394041422223242526272829303132333435363738  
192021 1 2 3 4 5 6 7 8 9101112131415161718 404142222324252627282930313233343536373839  
2021 1 2 3 4 5 6 7 8 910111213141516171819 414222232425262728293031323334353637383940  
21 1 2 3 4 5 6 7 8 91022121314151617181920 421123242526272829303132333435363738394041

222324252627282930313233343536373839404142 212019181716151413121110 9 8 7 6 5 4 3 2 1  
232425262728293031323334353637383940414222 2019181716151413121110 9 8 7 6 5 4 3 2 121

242526272829303132333435363738394041422223	19181716151413121110 9 8 7 6 5 4 3 2 12120
252627282930313233343536373839404142222324	181716151413121110 9 8 7 6 5 4 3 2 1212019
262728293031323334353637383940414222232425	1716151413121110 9 8 7 6 5 4 3 2 121201918
272829303132333435363738394041422223242526	16151413121110 9 8 7 6 5 4 3 2 12120191817
282930313233343536373839404142222324252627	151413121110 9 8 7 6 5 4 3 2 1212019181716
293031323334353637383940414222232425262728	1413121110 9 8 7 6 5 4 3 2 121201918171615
303132333435363738394041422223242526272829	13121110 9 8 7 6 5 4 3 2 12120191817161514
313233343536373839404111222324252627282930	12 610 9 8 7 2 5 4 318 1212019421716151413
323334353637383940412242232425262728293031	1110 9 8 7 6 5 4 3 1 221201918171615141312
333435363738394041224223242526272829303132	10 9 8 7 6 5 4 3 1 22120191817161514131211
343536373839404122422324252627282930313233	9 8 7 6 5 4 3 1 2212019181716151413121110
353637383940412242232425262728293031323334	8 7 6 5 4 3 1 2212019181716151413121110 9
363738394041421323242526272829303132333435	722 5 4 3 2 6212019 11716151418121110 9 8
373839404142222324252627282930313233343536	6 5 4 3 2 1212019181716151413121110 9 8 7
383940414222232425262728293031323334353637	5 4 3 2 1212019181716151413121110 9 8 7 6
394041422223242526272829303132333435363738	4 3 2 1212019181716151413121110 9 8 7 6 5
404142222324252627282930313233343536373839	3 2 1212019181716151413121110 9 8 7 6 5 4
414222232425262728293031323334353637383940	2 1212019181716151413121110 9 8 7 6 5 4 3
422223242526272829303132333435363738394041	1212019181716151413121110 9 8 7 6 5 4 3 2

LI(6,9) in S2 having pattren P6

1 2 3 4 5 6 7 8 9101112131415161718192021	222324252627282930313233343536373839404142
2 3 4 5 6 7 8 9101112131415161718192021 1	232425262728293031323334353637383940414222
3 4 5 6 7 8 9101112131415161718192021 1 2	242526272829303132333435363738394041422223
4 5 6 7 8 9101112131415161718192021 1 2 3	252627282930313233343536373839404142222324
5 6 7 8 9101112131415161718192021 1 2 3 4	262728293031323334353637383940414222232425
6 7 8 9101112134215161718192021 1 2 3 4 5	272829303132333435363738394041422223242526
7 8 9101112131415161718192021 1 2 3 4 5 6	282930313233343536373839404142222324252627
8 9101112131415161718192021 1 2 3 4 5 6 7	293031323334353637383940414222232425262728
9101112131415161718192021 1 2 3 4 5 6 7 8	303132333435363738394041422223242526272829
101112131415161718192021 1 2 3 4 5 6 7 8 9	313233343536373839404142222324252627282930
1112131415161718192021 1 2 3 4 5 6 7 8 910	323334353637383940414222232425262728293031
12131415161718192021 1 2 3 4 5 6 7 8 91011	333435363738394041422223242526272829303132
131415161718192021 1 2 3 4 5 6 7 8 9101112	343536373839404142222324252627282930313233
1415161718192021 1 2 3 4 5 6 7 8 910111213	353637383940414222232425262728293031323334
15161718192021 1 2 3 4 5 6 7 8 91011121314	363738394041422223242526272829303132333435
161718192021 1 2 3 4 5 6 7 8 9101112131415	373839404142222324252627282930313233343536
1718192021 1 2 3 4 5 6 7 8 910111213141516	383940414222232425262728293031323334353637
18192021 1 2 3 4 5 6 7 8 91011121314151617	394041422223242526272829303132333435363738
192021 1 2 3 4 5 6 7 8 9101112131415161718	404142222324252627282930313233343536373839
2021 1 2 3 4 5 6 7 8 910111213141516171819	414222232425262728293031323334353637383940
21 1 2 3 4 5 6 7 8 91022121314151617181920	421123242526272829303132333435363738394041

222324252627282930313233343536373839404142	212019181716151413121110 9 8 7 6 5 4 3 2 1
232425262728293031323334353637383940414222	2019181716151413121110 9 8 7 6 5 4 3 2 121
242526272829303132333435363738394041422223	19181716151413121110 9 8 7 6 5 4 3 2 12120
252627282930313233343536373839404142222324	181716151413121110 9 8 7 6 5 4 3 2 1212019
262728293031323334353637383940414222232425	1716151413121110 9 8 7 6 5 4 3 2 121201918
272829303132333435363738394041422223242526	16151413121110 9 8 7 6 5 4 3 2 12120191817
282930313233343536373839404142222324252627	151413121110 9 8 7 6 5 4 3 2 1212019181716
293031323334353637383940414222232425262728	1413121110 9 8 7 6 5 4 3 2 121201918171615
303132333435363738394041422223242526272829	13121110 9 8 7 6 5 4 3 2 12120191817161514
313233343536373839404111222324252627282930	12 710 9 8 3 6 5 4 1 218212019421716151413
323334353637383940412242232425262728293031	1110 9 8 7 6 5 4 3 2 121201918171615141312
333435363738394041224223242526272829303132	10 9 8 7 6 5 4 1 2 32120191817161514131211
343536373839404122422324252627282930313233	9 8 7 6 5 4 3 2 1212019181716151413121110
353637383940414214232425262728293031323334	822 6 5 4 7 2 3212019 11716151813121110 9
363738394041422223242526272829303132333435	7 6 5 4 3 2 1212019181716151413121110 9 8
373839404142222324252627282930313233343536	6 5 4 3 2 1212019181716151413121110 9 8 7

383940414222232425262728293031323334353637	5 4 3 2 1212019181716151413121110 9 8 7 6
394041422223242526272829303132333435363738	4 3 2 1212019181716151413121110 9 8 7 6 5
404142222324252627282930313233343536373839	3 2 1212019181716151413121110 9 8 7 6 5 4
414222232425262728293031323334353637383940	2 1212019181716151413121110 9 8 7 6 5 4 3
422223242526272829303132333435363738394041	1212019181716151413121110 9 8 7 6 5 4 3 2

LI(9,9) in S2 having pattren P7

1 2 3 4 5 6 7 8 9101112131415161718192021	222324252627282930313233343536373839404142
2 3 4 5 6 7 8 9101112131415161718192021 1	232425262728293031323334353637383940414222
3 4 5 6 7 8 9101112131415161718192021 1 2	242526272829303132333435363738394041422223
4 5 6 7 8 9101112131415161718192021 1 2 3	252627282930313233343536373839404142222324
5 6 7 8 9101112131415161718192021 1 2 3 4	262728293031323334353637383940414222232425
6 7 8 9101112131415161718192021 1 2 3 4 5	272829303132333435363738394041422223242526
7 8 9101112131415161718192021 1 2 3 4 5 6	282930313233343536373839404142222324252627
8 9101112131415161718192021 1 2 3 4 5 6 7	293031323334353637383940414222232425262728
9101112131415164218192021 1 2 3 4 5 6 7 8	303132333435363738394041172223242526272829
101112131415161718192021 1 2 3 4 5 6 7 8 9	313233343536373839404142222324252627282930
1112131415161718192021 1 2 3 4 5 6 7 8 910	323334353637383940414222232425262728293031
12131415161718192021 1 2 3 4 5 6 7 8 91011	333435363738394041422223242526272829303132
131415161718192021 1 2 3 4 5 6 7 8 9101112	343536373839404142222324252627282930313233
1415161718192021 1 2 3 4 5 6 7 8 910111213	353637383940414222232425262728293031323334
15161718192021 1 2 3 4 5 6 7 8 91011121314	363738394041422223242526272829303132333435
161718192021 1 2 3 4 5 6 7 8 9101112131415	373839404142222324252627282930313233343536
1718192021 1 2 3 4 5 6 7 8 910111213141516	383940414222232425262728293031323334353637
18192021 1 2 3 4 5 6 7 8 91011121314151617	394041422223242526272829303132333435363738
192021 1 2 3 4 5 6 7 8 9101112131415161718	404142222324252627282930313233343536373839
2021 1 2 3 4 5 6 7 8 910111213141516171819	414222232425262728293031323334353637383940
21 1 2 3 4 5 6 7 8 91022121314151617181920	421123242526272829303132333435363738394041
222324252627282930313233343536373839404142	212019181716151413121110 9 8 7 6 5 4 3 2 1
232425262728293031323334353637383940414222	2019181716151413121110 9 8 7 6 5 4 3 2 121
242526272829303132333435363738394041422223	19181716151413121110 9 8 7 6 5 4 3 2 12120
252627282930313233343536373839404142222324	181716151413121110 9 8 7 6 5 4 3 2 1212019
262728293031323334353637383940414222232425	1716151413121110 9 8 7 6 5 4 3 2 121201918
272829303132333435363738394041422223242526	16151413121110 9 8 7 6 5 4 3 2 12120191817
282930313233343536373839404142222324252627	151413121110 9 8 7 6 5 4 3 2 1212019181716
293031323334353637383940414222232425262728	1413121110 9 8 7 6 5 4 3 2 121201918171615
303132333435363738394041422223242526272829	13121110 9 8 7 6 5 4 3 2 12120191817161514
313233343536373839404111222324252627282930	12 710 9 8 4 6 521 3 2 1422019181716151413
323334353637383940412242232425262728293031	1110 9 8 7 6 5 3 4 2 121201918171615141312
333435363738394041224223242526272829303132	10 9 8 7 6 5 3 4 2 12120191817161514131211
343536373839404122422324252627282930313233	9 8 7 6 5 3 4 2 1212019181716151413121110
353637383940414217232425262728293031323334	822 6 5 4 7 2 1 32019182116151413121110 9
363738394041422223242526272829303132333435	7 6 5 4 3 2 1212019181716151413121110 9 8
373839404142222324252627282930313233343536	6 5 4 3 2 1212019181716151413121110 9 8 7
383940414222232425262728293031323334353637	5 4 3 2 1212019181716151413121110 9 8 7 6
394041422223242526272829303132333435363738	4 3 2 1212019181716151413121110 9 8 7 6 5
404142222324252627282930313233343536373839	3 2 1212019181716151413121110 9 8 7 6 5 4
414222232425262728293031323334353637383940	2 1212019181716151413121110 9 8 7 6 5 4 3
422223242526272829303132333435363738394041	1212019181716151413121110 9 8 7 6 5 4 3 2







LI(8,10) for mbc(2,23):

Pattern P6

First half of latin interchange

(AR)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5
7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6
8	9	10	11	12	13	14	15	16	46*18	19	20	21	22	23	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8
10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9
11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10
12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11
13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12
14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13
15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14
16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
23	1	2	3	4	5	6	7	8	9	10	11	24*13	14	15	16	17	18	19	20	21	22	

(lower BR)

24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27
29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29	30
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29	30	31
33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29	30	31	32
34	35	36	37	38	39	40	41	42	43	44	45	12*24	25	26	27	28	29	30	31	32	33	34
35	36	37	38	39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	36
36	37	38	39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	36	37
37	38	39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	36	37	38
38	39	40	41	42	43	44	45	46	17*25	26	27	28	29	30	31	32	33	34	35	36	37	38
39	40	41	42	43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
40	41	42	43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
41	42	43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
42	43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45

Second half of latin interchange

(upper BR)

24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24

26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25  
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26  
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27  
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28  
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29  
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 17\*24 25 26 27 28 29 30  
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31  
33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32  
34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33  
35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34  
36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35  
37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36  
38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37  
39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38  
40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39  
41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40  
42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41  
43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42  
44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43  
45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44  
46 12\*25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45

(SA)

23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  
22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23  
21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22  
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21  
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20  
18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19  
17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18  
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17  
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16  
14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16 15  
13 8\*11 10 9 4\* 7 6 5 2\* 3 21\* 1 23 22 46\*20 19 18 17 16 15 14  
12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13  
11 10 9 8 7 6 5 2\* 3 4\* 1 23 22 21 20 19 18 17 16 15 14 13 12  
10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11  
9 24\* 7 6 5 8\* 3 4\* 1 23 22 2\*20 19 18 21\*16 15 14 13 12 11 10  
8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9  
7 6 5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8  
6 5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7  
5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6  
4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5  
3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4  
2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3  
1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2

LI(8,8) for mbc(2,23):

Pattern P5

First half of latin interchange

(AR)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5
7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6
8	9	10	11	12	13	14	46*16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8
10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9
11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10
12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11
13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12
14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13
15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14
16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
23	1	2	3	4	5	6	7	8	9	10	11	24*13	14	15	16	17	18	19	20	21	22	

(lower BR)

24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27
29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29	30
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29	30	31
33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29	30	31	32
34	35	36	37	38	39	40	41	42	43	44	45	12*24	25	26	27	28	29	30	31	32	33	34
35	36	37	38	39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	36
36	37	38	39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	36	37
37	38	39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	36	37	38
38	39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
40	41	42	43	44	45	46	15*25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
42	43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45

Second half of latin interchange

(upper BR)

24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24  
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25  
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26  
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27  
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28  
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29  
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 15\*24 25 26 27 28 29 30  
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31  
33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32  
34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33  
35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34  
36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35  
37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36  
38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37  
39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38  
40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39  
41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40  
42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41  
43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42  
44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43  
45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44  
46 12\*25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45

(SA)  
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  
22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23  
21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22  
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21  
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20  
18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19  
17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18  
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17  
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16  
14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16 15  
13 6\*11 10 9 8 7 4\* 5 21\* 3 2 1 23 22 46\*20 19 18 17 16 15 14  
12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13  
11 10 9 8 7 6 5 2\* 3 4\* 1 23 22 21 20 19 18 17 16 15 14 13 12  
10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11  
9 8 7 6 5 4 3 23\* 1 2\*22 21 20 19 18 17 16 15 14 13 12 11 10  
8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9  
7 24\* 5 4 3 2 1 6\*22 23\*20 19 18 17 16 21\*14 13 12 11 10 9 8  
6 5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7  
5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6  
4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5  
3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4  
2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3  
1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2

LI(8,9) for mbc(2,23): Pattern P7

First half of latin interchange

(AR)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5
7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6
8	9	10	11	12	13	14	15	46*17	18	19	20	21	22	23	1	2	3	4	5	6	7	
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8
10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9
11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10
12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11
13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12
14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13
15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14
16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
23	1	2	3	4	5	6	7	8	9	10	11	24*13	14	15	16	17	18	19	20	21	22	

(lower BR)

24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27
29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29	30
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29	30	31
33	34	35	36	37	38	39	40	41	42	43	44	45	46	24	25	26	27	28	29	30	31	32
34	35	36	37	38	39	40	41	42	43	44	45	12*24	25	26	27	28	29	30	31	32	33	34
35	36	37	38	39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	
36	37	38	39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	36	
37	38	39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	36	37	
38	39	40	41	42	43	44	45	24*46*25	26	27	28	29	30	31	32	33	34	35	36	37	38	
39	40	41	42	43	44	45	46	16*25	26	27	28	29	30	31	32	33	34	35	36	37	38	
40	41	42	43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
41	42	43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
42	43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
43	44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
44	45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
45	46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
46	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45

Second half of latin interchange

(upper BR)

24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	24

26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25  
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26  
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27  
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28  
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29  
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 16\*24 25 26 27 28 29 30  
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31  
33 34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32  
34 35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33  
35 36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34  
36 37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35  
37 38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36  
38 39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37  
39 40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38  
40 41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39  
41 42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40  
42 43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41  
43 44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42  
44 45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43  
45 46 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44  
46 12\*25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45

(SA)  
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  
22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23  
21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22  
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21  
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20  
18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19  
17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18  
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17  
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16  
14 13 12 11 10 9 8 7 6 5 4 3 2 1 23 22 21 20 19 18 17 16 15  
13 7\*11 10 9 8 3\* 6 5 4 21\* 2 1 23 22 46\*20 19 18 17 16 15 14  
12 11 10 9 8 7 6 5 4 2\* 3\* 1 23 22 21 20 19 18 17 16 15 14 13  
11 10 9 8 7 6 5 4 2\* 3\* 1 23 22 21 20 19 18 17 16 15 14 13 12  
10 9 8 7 6 5 4 2\* 3\* 1 23 22 21 20 19 18 17 16 15 14 13 12 11  
9 8 7 6 5 4 2\* 3\* 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10  
8 24\* 6 5 4 3 7\* 1 23 22 2\*20 19 18 17 21\*15 14 13 12 11 10 9  
7 6 5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8  
6 5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7  
5 4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6  
4 3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5  
3 2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4  
2 1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3  
1 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2

## Latin Interchanges for mbc(2,25)

MBC (2, 25)

(AR)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5
7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6
8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9
11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10
12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11
13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12
14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13
15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14
16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
25	1																							

(lower BR)

(upper BR)

(SA)

25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24
22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21
19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20
18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16
14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15
13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14
12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13
11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12
10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6
4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5
3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4
2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
1	25																							





Second half of the latin interchange

(upper BR)

26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27
29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	14*26	27	28	29	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32
34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35
37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36
38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37
39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38
40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39
41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
50	13*27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	

(SA)

25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24
22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21
19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20
18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16
14	8*12	11	10	9	3*	7	6	5	4	24*	2	1	25	19*23	22	21	20	50*18	17	16	15			
13	12	11	10	9	8	7	6	5	4	2*	3*	1	25	24	23	22	21	20	19	18	17	16	15	14
12	11	10	9	8	7	6	5	4	3	1*	2*25	24	23	22	21	20	19	18	17	16	15	14	13	12
11	10	9	8	7	6	5	4	3	2	25*	1*24	23	22	21	20	19	18	17	16	15	14	13	12	11
10	9	8	7	6	5	4	3	2	1	24*25*23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
9	26*	7	6	5	4	8*	2	1	25	3*23	22	21	20	24*18	17	16	15	19*13	12	11	10			
8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6
4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5
3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4
2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2

First half of latin interchange

[illegible]

(lower BR)																																																	
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50																									
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26																									
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27																									
29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28																									
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29																									
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30																									
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31																									
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32																									
34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33																									
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34																									
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35																									
37	38	39	40	41	42	43	44	45	46	47	48	49	13*26	27	28	29	30	31	32	33	34	35	36	37																									
38	39	40	41	42	43	44	45	46	47	48	49	26*50*27	28	29	30	31	32	33	34	35	36	37	38	39																									
39	40	41	42	43	44	45	46	47	48	49	26*50*27	28	29	30	31	32	33	34	35	36	37	38	39	40																									
40	41	42	43	44	45	46	47	48	49	26*50*27	28	29	30	31	32	33	34	35	36	37	38	39	40	41																									
41	42	43	44	45	46	47	48	49	26*50*27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42																									
42	43	44	45	46	47	48	49	50	15*27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42																									
43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42																									
44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43																									
45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44																									
46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45																									
47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46																									
48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38																																		

# Second half of the latin interchange

(upper BR)

26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27
29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	15*26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32
34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35
37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36
38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37
39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38
40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39
41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
50	13*27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	

(SA)

25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24
22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21
19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20
18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16
14	8*12	11	10	9	3*	7	6	5	4	25*	2	1	20*24	23	22	21	50*19	18	17	16	15			
13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14
12	11	10	9	8	7	6	5	4	2*	1*	3*25	24	23	22	21	20	19	18	17	16	15	14	13	
11	10	9	8	7	6	5	4	3	25*	2*	1*24	23	22	21	20	19	18	17	16	15	14	13	12	
10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
9	26*	7	6	5	4	8*	2	1	3*24	23	22	21	25*19	18	17	16	20*14	13	12	11	10			
8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6
4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5
3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4
2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2



# Second half of the latin interchange

(upper BR)

26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50  
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 26  
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 26 27  
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 26 27 28  
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 26 27 28 29  
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 26 27 28 29 30  
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 14\*26 27 28 29 30 31  
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 26 27 28 29 30 31 32  
34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 26 27 28 29 30 31 32 33  
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 26 27 28 29 30 31 32 33 34  
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 26 27 28 29 30 31 32 33 34 35  
37 38 39 40 41 42 43 44 45 46 47 48 49 50 26 27 28 29 30 31 32 33 34 35 36  
38 39 40 41 42 43 44 45 46 47 48 49 50 26 27 28 29 30 31 32 33 34 35 36 37  
39 40 41 42 43 44 45 46 47 48 49 50 26 27 28 29 30 31 32 33 34 35 36 37 38  
40 41 42 43 44 45 46 47 48 49 50 26 27 28 29 30 31 32 33 34 35 36 37 38 39  
41 42 43 44 45 46 47 48 49 50 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40  
42 43 44 45 46 47 48 49 50 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41  
43 44 45 46 47 48 49 50 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42  
44 45 46 47 48 49 50 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43  
45 46 47 48 49 50 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44  
46 47 48 49 50 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45  
47 48 49 50 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46  
48 49 50 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47  
49 50 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48  
50 13\*27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49

(SA)

25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  
24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 25  
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 25 24  
22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 25 24 23  
21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 25 24 23 22  
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 25 24 23 22 21  
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 25 24 23 22 21 20  
18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 25 24 23 22 21 20 19  
17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 25 24 23 22 21 20 19 18  
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 25 24 23 22 21 20 19 18 17  
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 25 24 23 22 21 20 19 18 17 16  
14 6\*12 11 10 9 8 7 3\* 5 4 21\* 2 1 25 24 23 22 50\*20 19 18 17 16 15  
13 12 11 10 9 8 7 6 5 4 3 2 1 25 24 23 22 21 20 19 18 17 16 15 14  
12 11 10 9 8 7 6 5 4 2\* 1\* 3\*25 24 23 22 21 20 19 18 17 16 15 14 13  
11 10 9 8 7 6 5 4 2\* 3\*25\* 1\*24 23 22 21 20 19 18 17 16 15 14 13 12  
10 9 8 7 6 5 4 3 24\* 1\* 2\*25\*23 22 21 20 19 18 17 16 15 14 13 12 11  
9 8 7 6 5 4 3 2 1\*25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  
8 7 6 5 4 3 2 1 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9  
7 26\* 5 4 3 2 1 25 6\*23 22 24\*20 19 18 17 16 15 21\*13 12 11 10 9 8  
6 5 4 3 2 1 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7  
5 4 3 2 1 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6  
4 3 2 1 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5  
3 2 1 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4  
2 1 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3  
1 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2

LI(7,9) for MBC(2,25)

## First half of the latin interchange

(AR)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5
7	8	9	10	11	12	13	14	50*16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9
11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10
12	13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11
13	14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12
14	15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13
15	16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14
16	17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
19	20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
20	21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
21	22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
22	23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
23	24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
24	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
25	1																							

(lower BR)

26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27
29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32
34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35
37	38	39	40	41	42	43	44	45	46	47	48	49	13*26	27	28	29	30	31	32	33	34	35	36	37
38	39	40	41	42	43	44	45	46	47	48	49	26*50*27	28	29	30	31	32	33	34	35	36	37	38	39
39	40	41	42	43	44	45	46	47	48	49	26*50*27	28	29	30	31	32	33	34	35	36	37	38	39	40
40	41	42	43	44	45	46	47	48	49	26*50*27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
41	42	43	44	45	46	47	48	49	26*50*27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
42	43	44	45	46	47	48	49	26*50*27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
43	44	45	46	47	48	49	50	15*27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39								

## Second half of the latin interchange







# Second half of the latin interchange

(upper BR)

26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27
29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	15*26	27	28	29	30	31	32	
34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35
37	38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36
38	39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37
39	40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38
40	41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39
41	42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
42	43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
43	44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
44	45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
45	46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
46	47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
47	48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
48	49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
49	50	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
50	13*27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	

(SA)

25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24
22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21
19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20
18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16
14	6*12	11	10	9	8	7	4*	5	22*	3	2	1	25	24	23	50*21	20	19	18	17	16	15	14	
13	12	11	10	9	8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14
12	11	10	9	8	7	6	5	2*	3	4*	1	25	24	23	22	21	20	19	18	17	16	15	14	13
11	10	9	8	7	6	5	4	3	1*	2*25	24	23	22	21	20	19	18	17	16	15	14	13	12	
10	9	8	7	6	5	4	3	1*	2*25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	
9	8	7	6	5	4	3	2	24*25	1*23	22	21	20	19	18	17	16	15	14	13	12	11	10		
8	7	6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
7	26*	5	4	3	2	1	25	6*23	24*21	20	19	18	17	16	22*14	13	12	11	10	9	8			
6	5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
5	4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6
4	3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5
3	2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4
2	1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2

# Appendix C

## Bhaskar Rao Designs

### C.1 Thirteen inequivalent BRD(10, 5, 4)'s

BRD #	I	II	III	IV
Gibbons #	I	II	III	IV
1	$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$
2	$\begin{bmatrix} 1 & 2 & \bar{3} & 6 & 7 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & \bar{3} & 6 & 7 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{2} & 3 & 6 & 7 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & \bar{3} & 6 & 7 \end{bmatrix}$
3	$\begin{bmatrix} 1 & \bar{2} & \bar{3} & 8 & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{2} & 3 & 8 & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & \bar{3} & \bar{8} & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{2} & 3 & 8 & 9 \end{bmatrix}$
4	$\begin{bmatrix} 1 & \bar{2} & \bar{4} & \bar{5} & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{2} & 4 & 5 & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{2} & \bar{4} & 5 & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{2} & 4 & \bar{5} & 10 \end{bmatrix}$
5	$\begin{bmatrix} 1 & 3 & 6 & 7 & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{3} & \bar{6} & \bar{7} & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{3} & \bar{6} & \bar{8} & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{3} & \bar{6} & \bar{8} & 10 \end{bmatrix}$
6	$\begin{bmatrix} 1 & 4 & \bar{6} & 8 & \bar{9} \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{4} & 6 & 8 & \bar{9} \end{bmatrix}$	$\begin{bmatrix} 1 & 4 & \bar{6} & 7 & 8 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{4} & \bar{6} & 7 & \bar{9} \end{bmatrix}$
7	$\begin{bmatrix} 1 & \bar{4} & \bar{6} & \bar{8} & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{4} & \bar{6} & \bar{8} & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{4} & 6 & \bar{9} & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{4} & 6 & \bar{9} & \bar{10} \end{bmatrix}$
8	$\begin{bmatrix} 1 & 5 & \bar{7} & \bar{8} & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{5} & 7 & \bar{8} & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{5} & \bar{7} & \bar{8} & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & 5 & \bar{7} & \bar{8} & 9 \end{bmatrix}$
9	$\begin{bmatrix} 1 & \bar{5} & \bar{7} & \bar{9} & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{5} & \bar{7} & \bar{9} & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{5} & \bar{7} & \bar{9} & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & \bar{5} & \bar{7} & 8 & \bar{10} \end{bmatrix}$
10	$\begin{bmatrix} 2 & \bar{3} & 8 & 9 & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 2 & 3 & 8 & 9 & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 2 & 3 & \bar{7} & \bar{9} & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 2 & 3 & \bar{7} & \bar{9} & 10 \end{bmatrix}$
11	$\begin{bmatrix} 2 & \bar{4} & \bar{6} & 7 & \bar{9} \end{bmatrix}$	$\begin{bmatrix} 2 & 4 & \bar{6} & 7 & \bar{9} \end{bmatrix}$	$\begin{bmatrix} 2 & \bar{4} & \bar{6} & 7 & \bar{9} \end{bmatrix}$	$\begin{bmatrix} 2 & 4 & \bar{6} & 7 & 8 \end{bmatrix}$
12	$\begin{bmatrix} 2 & \bar{4} & \bar{7} & 8 & 10 \end{bmatrix}$	$\begin{bmatrix} 2 & \bar{4} & \bar{7} & 9 & 10 \end{bmatrix}$	$\begin{bmatrix} 2 & \bar{4} & 7 & \bar{8} & 10 \end{bmatrix}$	$\begin{bmatrix} 2 & \bar{4} & \bar{7} & 8 & 10 \end{bmatrix}$
13	$\begin{bmatrix} 2 & \bar{5} & 6 & \bar{7} & \bar{8} \end{bmatrix}$	$\begin{bmatrix} 2 & 5 & 6 & \bar{7} & \bar{8} \end{bmatrix}$	$\begin{bmatrix} 2 & 5 & 6 & \bar{8} & 9 \end{bmatrix}$	$\begin{bmatrix} 2 & \bar{5} & 6 & \bar{8} & 9 \end{bmatrix}$
14	$\begin{bmatrix} 2 & \bar{5} & \bar{6} & 9 & 10 \end{bmatrix}$	$\begin{bmatrix} 2 & \bar{5} & \bar{6} & 8 & 10 \end{bmatrix}$	$\begin{bmatrix} 2 & \bar{5} & 6 & 8 & 10 \end{bmatrix}$	$\begin{bmatrix} 2 & \bar{5} & \bar{6} & 9 & \bar{10} \end{bmatrix}$
15	$\begin{bmatrix} 3 & 4 & \bar{5} & \bar{6} & 9 \end{bmatrix}$	$\begin{bmatrix} 3 & \bar{4} & 5 & \bar{6} & \bar{9} \end{bmatrix}$	$\begin{bmatrix} 3 & \bar{4} & \bar{5} & \bar{6} & 9 \end{bmatrix}$	$\begin{bmatrix} 3 & \bar{4} & \bar{5} & \bar{6} & \bar{8} \end{bmatrix}$
16	$\begin{bmatrix} 3 & \bar{4} & 5 & \bar{7} & 8 \end{bmatrix}$	$\begin{bmatrix} 3 & 4 & \bar{5} & \bar{7} & \bar{8} \end{bmatrix}$	$\begin{bmatrix} 3 & 4 & \bar{5} & 7 & \bar{8} \end{bmatrix}$	$\begin{bmatrix} 3 & \bar{4} & 5 & 7 & 9 \end{bmatrix}$
17	$\begin{bmatrix} 3 & \bar{4} & 7 & 9 & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 3 & \bar{4} & 7 & \bar{8} & 10 \end{bmatrix}$	$\begin{bmatrix} 3 & \bar{4} & 8 & 9 & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 3 & 4 & \bar{8} & \bar{9} & \bar{10} \end{bmatrix}$
18	$\begin{bmatrix} 3 & \bar{5} & 6 & 8 & \bar{10} \end{bmatrix}$	$\begin{bmatrix} 3 & \bar{5} & 6 & \bar{9} & 10 \end{bmatrix}$	$\begin{bmatrix} 3 & 5 & \bar{6} & \bar{7} & 10 \end{bmatrix}$	$\begin{bmatrix} 3 & \bar{5} & 6 & 7 & 10 \end{bmatrix}$

BRD #	V					VI					VII					VIII				
Gibbons #	V					VI					VII					VIII				
1	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
2	1	2	$\bar{3}$	6	7	1	$\bar{2}$	3	6	7	1	$\bar{2}$	3	6	7	1	2	$\bar{3}$	6	7
3	1	$\bar{2}$	3	8	9	1	$\bar{2}$	$\bar{3}$	8	9	1	$\bar{2}$	$\bar{3}$	8	9	1	$\bar{2}$	3	8	9
4	1	$\bar{2}$	4	5	10	1	2	$\bar{4}$	5	10	1	2	$\bar{4}$	$\bar{5}$	10	1	$\bar{2}$	$\bar{4}$	5	10
5	1	$\bar{3}$	$\bar{6}$	$\bar{7}$	10	1	$\bar{3}$	$\bar{6}$	$\bar{8}$	$\bar{10}$	1	$\bar{3}$	6	$\bar{8}$	10	1	$\bar{3}$	$\bar{6}$	$\bar{8}$	$\bar{10}$
6	1	$\bar{4}$	$\bar{6}$	$\bar{8}$	$\bar{10}$	1	4	6	$\bar{7}$	$\bar{8}$	1	4	$\bar{6}$	7	9	1	4	6	$\bar{7}$	10
7	1	$\bar{4}$	7	8	$\bar{9}$	1	$\bar{4}$	$\bar{7}$	$\bar{9}$	$\bar{10}$	1	$\bar{4}$	$\bar{7}$	$\bar{8}$	$\bar{10}$	1	$\bar{4}$	8	$\bar{9}$	$\bar{10}$
8	1	$\bar{5}$	6	$\bar{8}$	9	1	$\bar{5}$	$\bar{6}$	9	10	1	$\bar{5}$	$\bar{6}$	$\bar{9}$	$\bar{10}$	1	$\bar{5}$	$\bar{6}$	$\bar{7}$	$\bar{9}$
9	1	$\bar{5}$	$\bar{7}$	$\bar{9}$	$\bar{10}$	1	$\bar{5}$	7	8	$\bar{9}$	1	5	$\bar{7}$	8	$\bar{9}$	1	$\bar{5}$	7	$\bar{8}$	9
10	2	3	8	9	$\bar{10}$	2	$\bar{3}$	7	9	$\bar{10}$	2	$\bar{3}$	7	$\bar{9}$	10	2	3	$\bar{7}$	9	$\bar{10}$
11	2	4	$\bar{6}$	7	$\bar{8}$	2	4	$\bar{6}$	7	$\bar{9}$	2	4	6	$\bar{7}$	8	2	$\bar{4}$	6	$\bar{7}$	$\bar{8}$
12	2	$\bar{4}$	$\bar{6}$	9	10	2	$\bar{4}$	6	8	$\bar{10}$	2	$\bar{4}$	6	9	$\bar{10}$	2	$\bar{4}$	$\bar{6}$	8	9
13	2	5	6	$\bar{7}$	$\bar{9}$	2	$\bar{5}$	6	$\bar{8}$	9	2	5	$\bar{6}$	$\bar{8}$	9	2	5	$\bar{6}$	$\bar{9}$	10
14	2	$\bar{5}$	$\bar{7}$	8	10	2	$\bar{5}$	$\bar{7}$	8	10	2	$\bar{5}$	7	8	$\bar{10}$	2	$\bar{5}$	7	8	10
15	3	4	$\bar{5}$	$\bar{6}$	$\bar{9}$	3	$\bar{4}$	5	$\bar{6}$	9	3	4	$\bar{5}$	6	$\bar{8}$	3	$\bar{4}$	$\bar{5}$	6	$\bar{9}$
16	3	$\bar{4}$	5	$\bar{7}$	$\bar{8}$	3	$\bar{4}$	$\bar{5}$	7	$\bar{8}$	3	$\bar{4}$	5	7	$\bar{9}$	3	$\bar{4}$	5	7	$\bar{8}$
17	3	$\bar{4}$	7	$\bar{9}$	10	3	4	8	9	$\bar{10}$	3	$\bar{4}$	8	9	10	3	4	7	$\bar{9}$	$\bar{10}$
18	3	$\bar{5}$	6	$\bar{8}$	10	3	$\bar{5}$	$\bar{6}$	$\bar{7}$	$\bar{10}$	3	$\bar{5}$	$\bar{6}$	$\bar{7}$	10	3	$\bar{5}$	$\bar{6}$	$\bar{8}$	10

BRD #	X	XI	XVII	XVIII
Gibbons #	X	XI	XVII	XVIII
1	[ 1 2 3 4 5 ]	[ 1 2 3 4 5 ]	[ 1 2 3 4 5 ]	[ 1 2 3 4 5 ]
2	[ 1 2 $\bar{3}$ 6 7 ]	[ 1 $\bar{2}$ $\bar{3}$ 6 7 ]	[ 1 $\bar{2}$ 3 6 7 ]	[ 1 $\bar{2}$ $\bar{3}$ 6 7 ]
3	[ 1 $\bar{2}$ 3 8 9 ]	[ 1 $\bar{2}$ 3 8 9 ]	[ 1 $\bar{2}$ $\bar{3}$ 8 9 ]	[ 1 2 $\bar{5}$ 7 8 ]
4	[ 1 $\bar{2}$ $\bar{4}$ 5 10 ]	[ 1 2 $\bar{4}$ $\bar{6}$ 10 ]	[ 1 2 $\bar{4}$ $\bar{6}$ 10 ]	[ 1 $\bar{2}$ $\bar{6}$ $\bar{7}$ 9 ]
5	[ 1 $\bar{3}$ $\bar{6}$ $\bar{8}$ 10 ]	[ 1 $\bar{3}$ 5 $\bar{8}$ 10 ]	[ 1 $\bar{3}$ $\bar{5}$ $\bar{8}$ 10 ]	[ 1 3 $\bar{5}$ 9 10 ]
6	[ 1 4 6 $\bar{9}$ $\bar{10}$ ]	[ 1 4 $\bar{5}$ 6 9 ]	[ 1 $\bar{4}$ 5 $\bar{9}$ $\bar{10}$ ]	[ 1 $\bar{3}$ $\bar{6}$ 8 $\bar{10}$ ]
7	[ 1 $\bar{4}$ $\bar{7}$ 8 $\bar{10}$ ]	[ 1 $\bar{4}$ $\bar{7}$ $\bar{8}$ $\bar{10}$ ]	[ 1 4 $\bar{6}$ 7 $\bar{9}$ ]	[ 1 $\bar{4}$ 7 $\bar{8}$ 10 ]
8	[ 1 $\bar{5}$ $\bar{6}$ 7 $\bar{9}$ ]	[ 1 $\bar{5}$ $\bar{7}$ 8 $\bar{9}$ ]	[ 1 $\bar{5}$ 6 $\bar{7}$ $\bar{8}$ ]	[ 1 $\bar{4}$ $\bar{7}$ 9 $\bar{10}$ ]
9	[ 1 $\bar{5}$ $\bar{7}$ $\bar{8}$ 9 ]	[ 1 $\bar{6}$ 7 $\bar{9}$ $\bar{10}$ ]	[ 1 $\bar{7}$ 8 9 $\bar{10}$ ]	[ 1 5 6 $\bar{8}$ $\bar{9}$ ]
10	[ 2 3 7 9 10 ]	[ 2 $\bar{3}$ 7 9 $\bar{10}$ ]	[ 2 $\bar{3}$ 7 9 $\bar{10}$ ]	[ 2 $\bar{3}$ $\bar{7}$ $\bar{8}$ 9 ]
11	[ 2 $\bar{4}$ 6 $\bar{7}$ $\bar{8}$ ]	[ 2 4 $\bar{5}$ 7 $\bar{9}$ ]	[ 2 $\bar{4}$ $\bar{5}$ 7 8 ]	[ 2 $\bar{3}$ 8 $\bar{9}$ 10 ]
12	[ 2 $\bar{4}$ $\bar{6}$ 8 $\bar{9}$ ]	[ 2 $\bar{4}$ 8 9 $\bar{10}$ ]	[ 2 4 $\bar{7}$ 8 10 ]	[ 2 4 $\bar{5}$ $\bar{8}$ $\bar{10}$ ]
13	[ 2 5 $\bar{6}$ 9 $\bar{10}$ ]	[ 2 $\bar{5}$ 6 $\bar{7}$ $\bar{8}$ ]	[ 2 5 6 $\bar{8}$ 9 ]	[ 2 $\bar{4}$ 6 9 $\bar{10}$ ]
14	[ 2 $\bar{5}$ $\bar{7}$ 8 10 ]	[ 2 5 6 8 10 ]	[ 2 $\bar{5}$ 6 $\bar{9}$ $\bar{10}$ ]	[ 2 5 $\bar{6}$ $\bar{7}$ 10 ]
15	[ 3 $\bar{4}$ $\bar{5}$ 6 7 ]	[ 3 $\bar{4}$ $\bar{5}$ 7 10 ]	[ 3 4 $\bar{5}$ 7 9 ]	[ 3 4 $\bar{5}$ $\bar{7}$ $\bar{9}$ ]
16	[ 3 $\bar{4}$ 5 $\bar{8}$ $\bar{9}$ ]	[ 3 $\bar{4}$ 6 7 $\bar{8}$ ]	[ 3 $\bar{4}$ $\bar{6}$ $\bar{8}$ 9 ]	[ 3 $\bar{4}$ 6 $\bar{7}$ 8 ]
17	[ 3 4 $\bar{7}$ $\bar{9}$ 10 ]	[ 3 4 $\bar{6}$ $\bar{8}$ 9 ]	[ 3 $\bar{4}$ 6 8 10 ]	[ 3 5 $\bar{6}$ 7 $\bar{10}$ ]
18	[ 3 $\bar{5}$ $\bar{6}$ $\bar{8}$ $\bar{10}$ ]	[ 3 5 6 $\bar{9}$ $\bar{10}$ ]	[ 3 $\bar{5}$ $\bar{6}$ $\bar{7}$ $\bar{10}$ ]	[ 4 5 6 8 9 ]

BRD #	XX
Gibbons #	XX
1	1 2 3 4 5
2	1 2 $\bar{3}$ $\bar{4}$ 6
3	1 $\bar{2}$ 5 7 8
4	1 $\bar{2}$ $\bar{6}$ $\bar{7}$ 9
5	1 3 $\bar{5}$ $\bar{9}$ 10
6	1 $\bar{3}$ 7 9 10
7	1 4 6 $\bar{8}$ $\bar{9}$
8	1 $\bar{4}$ $\bar{7}$ 8 $\overline{10}$
9	1 $\bar{5}$ $\bar{6}$ $\bar{8}$ $\overline{10}$
10	2 3 $\bar{6}$ 8 10
11	2 $\bar{3}$ $\bar{7}$ 8 $\bar{9}$
12	2 4 5 9 $\overline{10}$
13	2 $\bar{4}$ $\bar{8}$ 9 10
14	2 $\bar{5}$ $\bar{6}$ 7 $\overline{10}$
15	3 $\bar{4}$ 5 $\bar{7}$ $\bar{8}$
16	3 $\bar{4}$ 6 7 $\overline{10}$
17	3 $\bar{5}$ 6 8 9
18	4 $\bar{5}$ 6 $\bar{7}$ 9

## Appendix D

---

### Source Code for Room Squares, Latin Squares and BRDs

```

#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <iomanip.h>
#include "../Cell.h"
#include "RoomSquare.h"

#define DEBUGBACKUP
#define DEBUGREAD
#define DEBUGFIND
#define DEBUGDEL
#define DEBUGCOMP
#define DEBUGEQUAL
#define DEBUGPAIRS
#define DEBUGRCCELLMOVE

#define DEBUGESCS
#define DEBUGELCS
#define DEBUGELCSSAVED

void Usage(char *fname){
    cout << "Usage is: " << fname << " -cs      SeedValue MatrixFileName\n";
    cout << "or      : " << fname << " -complete SeedValue MatrixFileName\n";
    cout << "or      : " << fname <<
    " -slbcs      SeedValue MatrixFileName BackupFileName\n";
    cout << "or      : " << fname <<
    " -scs      SeedValue MatrixFileName BackupFileName StartSize\n";
    cout << "or      : " << fname <<
    " -lcs      SeedValue MatrixFileName BackupFileName StartSize\n";
    cout << "or      : " << fname <<
    -;
    cout << "or      : " << fname <<
    " -rlcs      SeedValue MatrixFileName BackupFileName StartSize\n";
    cout <<
    "\nNote that if you already know a critical of size X, StartSize should be X-1\n";
}

main(int argc, char *argv[]){
    char *option;
    long SeedValue;
    char *MatrixFileName, *BackupFileName;
    int StartSize;
    RoomSquare square;

    if(argc <= 3)
        Usage(argv[0]);
    else{
        option = argv[1];
        sscanf(argv[2], "%d", &SeedValue);          // Seed for RNG
        srand48(SeedValue);                          // Randomize the RNG
        MatrixFileName = argv[3];
        if(square.ReadMatrix(MatrixFileName, CheckIt)){
            // Read matrix (CheckIt or NoCheck)
            if(strcmp(option, "-cs") == 0){
                if(square.FindCriticalSet())
                    cout << "Critical Set was found.\n";
                else
                    cout << "Couldn't find a Critical Set!\n";
            }
            else if(strcmp(option, "-complete") == 0){
                square.Complete();
            }
            // Complete a partial room square
            else if(strcmp(option, "-slbcs") == 0){
                BackupFileName = argv[4];
                square.ApproxCSBounds(BackupFileName); // Find Bounds
            }
            // for Critical Sets
            else if(strcmp(option, "-scs") == 0){
                if(argc <= 5)
                    Usage(argv[0]);
                else{
                    // Set matrix
                    BackupFileName = argv[4];
                    sscanf(argv[5], "%d", &StartSize);
                }
            }
        }
    }
}

```



// Start size

```

        if
(square.FindSmallestCriticalSet(BackupFileName, StartSize)) // Find Critical Set
        cout << "Critical Set was found.\n";
        else
        cout << "Couldn't find a Critical Set!\n"

    }
}
else if(strcmp(option, "-lcs") == 0){
    if(argc <= 5)
        Usage(argv[0]);
    else{ // Set matrix
        BackupFileName = argv[4];
        sscanf(argv[5], "%d", &StartSize);
        if
(square.FindLargestCriticalSet(BackupFileName, StartSize)) // Find Critical Set
        cout << "Critical Set was found.\n";
        else
        cout << "Couldn't find a Critical Set!\n"

    }
}
else if(strcmp(option, "-rscs") == 0){
    if(argc <= 5)
        Usage(argv[0]);
    else{ // Set matrix
        BackupFileName = argv[4];
        sscanf(argv[5], "%d", &StartSize);
// Start size
        if
(square.FindRandomSmallestCriticalSet(BackupFileName, StartSize))
// Find Critical Set
        cout << "Critical Set was found.\n";
        else
        cout << "Couldn't find a Critical Set!\n"

    }
}
else if(strcmp(option, "-rlcs") == 0){
    if(argc <= 5)
        Usage(argv[0]);
    else{ // Set matrix
        BackupFileName = argv[4];
        sscanf(argv[5], "%d", &StartSize);
// Start size
        if
(square.FindRandomLargestCriticalSet(BackupFileName, StartSize))
// Find Critical Set
        cout << "Critical Set was found.\n";
        else
        cout << "Couldn't find a Critical Set!\n"

    }
}
else
    Usage(argv[0]);
}
}

```

RoomSquare::RoomSquare(){

```

    r = 0;
    initialized = false;
    deleted = false;
    partial = false;
    foundsmaller = false;
    foundlarger = false;

```

RoomSquare::~RoomSquare(){

```

    int i;

    if(initialized){
        for(i=0; i<r; i++){
            delete [] Matrix[i];
            delete [] MatrixTemp[i];

```

```

        delete [] MatrixOrig[i];
        delete [] MatrixSaved[i];
    }
    delete [] Matrix;
    delete [] MatrixTemp;
    delete [] MatrixOrig;
    delete [] MatrixSaved;
    delete [] RCells;
    delete [] KnownNumbers;
    delete [] RemainNumbers;
}

bool RoomSquare::ReadMatrix(char *InFileName, bool check){
    int i, j;
    int e1, e2;
    bool ok = true;
    ifstream file;

    file.open(InFileName, ios::in);
    if(file.good()){
        file >> r;
        if(file.good() && Init()){
            for(i=0; (i<r) && ok; i++){
                for(j=0; (j<r) && ok; j++){
                    file >> e1 >> e2;
                    if(file.fail()){
                        ok = false;
                    }
                    else if( (e1 == UNKNOWN) && (e2 == UNKNOWN) ){
                        Matrix[i][j].SetUnKnown();
                        partial = true;
                    }
                    else if( (e1 == EMPTY) && (e2 == EMPTY) ){
                        Matrix[i][j].SetEmpty();
                    }
                    else{
                        ok = Matrix[i][j].SetValue(e1, e2);
                    }
                }
            }
            for(i=0; (i<r) && ok && partial; i++){ // Read the complete one
                if previously a partial one entered
                    for(j=0; (j<r) && ok; j++){
                        file >> e1 >> e2;
                        if(file.fail()){
                            ok = false;
                        }
                        else if( (e1 == UNKNOWN) && (e2 == UNKNOWN) ){
                            cerr <<
                                "Second matrix should not have any unknown element!\n";
                            ok = false;
                        }
                        else if( (e1 == EMPTY) && (e2 == EMPTY) ){
                            MatrixOrig[i][j].SetEmpty();
                        }
                        else{
                            ok = MatrixOrig[i][j].SetValue(e1, e2);
                        }
                    }
                }
            }
        }
        else
            ok = false;
        file.close();
    }
    else ok = false;

#ifdef DEBUGREAD
    cout << "Test started\n";
    PrintMatrix(Matrix);
    PrintRCells();
    cout << "Test finished\n";
#endif

    if(ok){
        if(!partial){

```

```

        delete [] MatrixOrig[i];
        delete [] MatrixSaved[i];
    }
    delete [] Matrix;
    delete [] MatrixTemp;
    delete [] MatrixOrig;
    delete [] MatrixSaved;
    delete [] RCells;
    delete [] KnownNumbers;
    delete [] RemainNumbers;
}

bool RoomSquare::ReadMatrix(char *InFileName, bool check){
    int i, j;
    int e1, e2;
    bool ok = true;
    ifstream file;

    file.open(InFileName, ios::in);
    if(file.good()){
        file >> r;
        if(file.good() && Init()){
            for(i=0; (i<r) && ok; i++){
                for(j=0; (j<r) && ok; j++){
                    file >> e1 >> e2;
                    if(file.fail()){
                        ok = false;
                    }
                    else if( (e1 == UNKNOWN) && (e2 == UNKNOWN) ){
                        Matrix[i][j].SetUnKnown();
                    }
                    partial = true;
                }
                else if( (e1 == EMPTY) && (e2 == EMPTY) ){
                    Matrix[i][j].SetEmpty();
                }
                else{
                    ok = Matrix[i][j].SetValue(e1, e2);
                }
            }
        }
        for(i=0; (i<r) && ok && partial; i++){ // Read the complete one
            if previously a partial one entered
                for(j=0; (j<r) && ok; j++){
                    file >> e1 >> e2;
                    if(file.fail()){
                        ok = false;
                    }
                    else if( (e1 == UNKNOWN) && (e2 == UNKNOWN) ){
                        cerr <<
                            "Second matrix should not have any unknown element!\n";
                        ok = false;
                    }
                    else if( (e1 == EMPTY) && (e2 == EMPTY) ){
                        MatrixOrig[i][j].SetEmpty();
                    }
                    else{
                        ok = MatrixOrig[i][j].SetValue(e1, e2);
                    }
                }
            }
        }
    }
    else
        ok = false;
    file.close();
}
else ok = false;

#ifdef DEBUGREAD
    cout << "Test started\n";
    PrintMatrix(Matrix);
    PrintRCells();
    cout << "Test finished\n";
#endif

if(ok){
    if(!partial){

```

```
CopyMatrix(MatrixOrig); 206
```

```

    }
    ok = check ? CheckAccuracy() : true;
}
else{ 210
    cerr << "Cannot work with input file.\n";
}

if(ok){
    cout << "Reading the matrix is successfully completed.\n\n";
}
cout.flush();

return ok; 220
}

```

```

bool RoomSquare::CannotDelete(int start){
    bool cannot = true;
    int i;

    for(i=0; i<start; i++){
        if(DeletionTry(RCells[i])){           // Try to delete the cell
            cannot = false;
            break;
        } 230

    }

    return cannot;
}

```

```

bool RoomSquare::FindCriticalSet(){
    DeleteAll();           // Try each element (randomly) and try to delete it
    PrintMatrix(Matrix);   // Display the matrix after deletion trials.
    return deleted;        // true if at least one cell is deleted
}

```

```

void RoomSquare::DeleteAll(){
    long i, j, index;

    NoOfRCells = 0;
    for(i=0; i<r; i++){
        for(j=0; j<r; j++){
            RCells[NoOfRCells++] = &Matrix[i][j];    // or Matrix[i] + j
        }
    }

    while(NoOfRCells != 0){
        index = lrand48() % NoOfRCells;

        // Remove the unwanted cells. They are either necessary cells or the ones
        // that can be deleted (and reconstructed again).
        DeletionTry(RCells[index]);           // Try to delete the cell
        RCells[index] = RCells[--NoOfRCells]; // Remove this cell from the list

#ifdef DEBUGFIND
        PrintRCells(); 260
#endif
    }
}

```

```

void RoomSquare::MakeBackup(char *BackupFileName, int num){ ...
//num is the number of known cells (start)
    int i, row, col;
    ofstream file;
    char tmpstr[1000];

    sprintf(tmpstr, "mv %s %s.bak", BackupFileName, BackupFileName);
    system(tmpstr); 270
    file.open(BackupFileName, ios::out | ios::trunc);
    if(file.good()){
        file << num << "\n";
        if(file.good()){
            for(i=0; i<num; i++){
                RCells[i]->GetRowCol(row,col);
                file << row << " " << col << "\n";
                if(file.fail()){
                    cerr << "Error -- Cannot write to backup file.\n" ...
                }
            }
            break; 280
        }
    }
}

```

```

    }
    }
    }
    else
        cerr << "Error -- Cannot write to backup file.\n";
        file.close();
    }
    else
        cerr << "Error -- Cannot create backup file.\n";
}
290

bool RoomSquare::MakeApproxCSBoundsBackup(char *BackupFileName, int iter, int
LowestSize, int Lseed, int HighestSize, int Hseed){
    bool ok = true;
    ofstream file;
    char tmpstr[1000];

    sprintf(tmpstr, "mv %s %s.bak", BackupFileName, BackupFileName);
    system(tmpstr);
    file.open(BackupFileName, ios::out | ios::trunc);
    if(file.good()){
300

        file << iter << endl
            << LowestSize << ' ' << Lseed << endl
            << HighestSize << ' ' << Hseed << endl
            <<
            "\n\nThe following are current approximations on size of CSs after " << iter <<
            " iterations:\n"
            << "Lowest CS size: " << LowestSize << ", when seed="
            << Lseed << endl
            << "Highest CS size: " << HighestSize << ", when seed="
            << Hseed << endl;

        if(file.fail()){
            cerr << "Error -- Cannot write to backup file.\n";
            ok = false;
        }
        file.close();
    }
    else{
        cerr << "Error -- Cannot create backup file.\n";
        ok = false;
    }

    return ok;
320
}

int RoomSquare::ReadBackup(char *BackupFileName, int num){
    // num is number of known cells (start)
    int i, row, col, start;
    ifstream file;

    start = num;
    file.open(BackupFileName, ios::in);
    if(file.good()){
330
        file >> start;
        if(file.good()){
            for(i=0; i<start ; i++){
                file >> row >> col;
                if(file.fail()){
                    cerr << "Error -- Cannot read backup file.\n";
                    exit(1);
                }
                RCells[i] = &Matrix[row][col];
            }
            cout << "BACKUP: Program continues from the previous point ("
            -")\n";
            cout.flush();
        }
        else{
            cerr << "Cannot read from backup file.\n";
            exit(1);
        }
        file.close();
    }
    // Otherwise, this is the first time, and there is no backup!

#ifdef DEBUGBACKUP

```

```

    cout << "----- Reading Backup file -----\\n";
    cout << "start = " << num << "\\n";
    for(i=0; i<num ; i++){
        RCells[i]->GetRowCol(row,col);
        cout << "RCell[" << row << "]" << col << "]"\\n";
    }
    cout << "----- Reading Backup file Finished -----\\n";
#endif

    return start;
}

void RoomSquare::ReadApproxCSBoundsBackup(char *BackupFileName, int &iter, int
&LowestSize, int &Lseed, int &HighestSize, int &Hseed){
    ifstream file;

    file.open(BackupFileName, ios::in);
    if(file.good()){
        file >> iter >> LowestSize >> Lseed >> HighestSize >> Hseed;
        if(file.fail()){
            cerr << "Cannot read from backup file.\\n";
            exit(1);
        }
        file.close();
    }
    else{
        cout <<
"Couldn't open backup file. We assume that this is the first time, and there is no backup!\\n";
        cout.flush();
    }
}

bool RoomSquare::FindLargestCriticalSet(char *BackupFileName, int start){
    bool done = false;
    int temp, count=0;          // Counter for backup

    // RCells are used here to point to the celles that are known
    // We only use 'start' of them

    InitializeRCells(start);      // To the first start elements
    if((temp = ReadBackup(BackupFileName, start)) < start){
        cout << "In backup: start=" << temp << ".\\nProgram uses the entered value: start=" << start << "
        InitializeRCells(start);      // To the first start elements
    }
    else{
        start = temp;
        cout << "Program starts by:  start=" << start << ".\\n";
    }
    cout.flush();
    do{
        SetAllUnKnown(Matrix);      // Set all elements to unknown
        SetKnownElements(start);
        // Based of RCells[0] to RCells[start-1] set start elements
#ifdef DEBUGELCS
        PrintMatrix(Matrix);
        cout.flush();
#endif
        if(CanComplete() && CannotDelete(start)){
            foundlarger = true;      // At least a CS of this size found
            cout << "A critical set of size " << start << " is found.\\n";
            PrintMatrix(Matrix);
            start++;                  // Try to find an smaller one
            InitializeRCells(start);  // To the first start elements
            system("date");
            cout << "Now looking for size " << start << " ...\\n";
            cout.flush();
        }
        else{
            done = MoveRCellsForward(start);
            if(count++ == MAXBACKUP){
                MakeBackup(BackupFileName, start);
                count = 0;
            }
        }
    }
}

```

```

    }
    }while(!done);
    cout << "Failed to find a critical set of size " << start << ", after exhaustively searching the sp
    return foundlarger;    // true if at least one cell is deleted
}

bool RoomSquare::FindSmallestCriticalSet(char *BackupFileName, int start){
    bool done = false;
    int temp, count=0;    // Counter for backup

    // RCells are used here to point to the celles that are known
    // We only use 'start' of them

    InitializeRCells(start);    // To the first start elements
    if((temp = ReadBackup(BackupFileName, start)) > start){
        cout << "In backup: start=" << temp <<
        "\nProgram uses the entered value: start=" << start <<
        ", but continues from backup point.\n";
    }
    else{
        start = temp;
        cout << "Program starts by:  start=" << start << ".\n";
    }
    cout.flush();
    do{
        SetAllUnKnown(Matrix);    // Set all elements to unknown
        SetKnownElements(start);
    // Based of RCells[0] to RCells[start-1] set start elements

#ifdef DEBUGESCS
        PrintMatrix(Matrix);
        cout.flush();
#endif
        if(CanComplete()){    // ... if the matrix can be completed.
            foundsmaller = true;    // At least a CS of this size found
            cout << "A matrix of size " << start <<
            " is found that can uniquely complete RS.\n";
            PrintMatrix(Matrix);
            start--;    // Try to find an smaller one
            count = 0;
            // We don't use InitializeRCells(start), because we want to
            // continue from where we finished with previous start value.
            system("date");
            cout << "Now looking for size " << start << " ...\n";
            cout.flush();
        }
        else{
            done = MoveRCellsForward(start);
    // Next possibility (exhaustive search)
            if(count++ == MAXBACKUP){
                MakeBackup(BackupFileName, start);
                count = 0;
            }
        }
    }while(!done);
    cout << "Failed to find a critical set of size " << start << ", after exhaustively searching the sp
    return foundsmaller;    // true if at least one cell is deleted
}

bool RoomSquare::FindRandomLargestCriticalSet(char *BackupFileName, int start){
    int temp, count1=0, count2=0, savedstart=0;    // Counter for backup

    // RCells are used here to point to the celles that are known
    // We only use 'start' of them

    if((temp = ReadBackup(BackupFileName, start)) < start){
        cout << "RAND: In backup: start=" << temp <<
        "\nProgram uses the entered value: start=" << start << ".\n";
        InitializeRCells(start);    // To the first start elements
    }
    else{
        start = temp;
        cout << "RAND: Program starts by:  start=" << start << ".\n";
    }
}

```

```

    cout.flush();
    for(;;){
        RandomlySelectRCells(start);
// Randomly select RCells[0] to RCells[start-1]

#ifdef DEBUGELCS
    cout << "Every loop:\n"; PrintMatrix(Matrix); cout.flush();
#endif
    while(!CanComplete()){
        AddAnotherRandomCell(start);

#ifdef DEBUGELCS
        cout << "One added:\n"; PrintMatrix(Matrix); cout.flush();
#endif
        start++;
    }
    if(CannotDelete(start)){ // ... if the matrix can be completed.
        foundlarger = true; // At least a CS of this size found
        cout << "A critical set of size " << start << " is found.\n";
        PrintMatrix(Matrix);
        start++; // Try to find an smaller one
        count1 = 0;
        count2 = 0;
        system("date");
        cout << "Now looking for size " << start << " ...\n";
        cout.flush();
    }
    else{
        if(savedstart < start){
// If this is the first matrix of this size
            CopyMatrix(MatrixSaved);
            savedstart = start;

#ifdef DEBUGELCSSAVED
            cout << "Matrix saved:\n"
; PrintMatrix(Matrix); cout.flush();
#endif
        }
        if(count1++ == MAXBACKUP){
            MakeBackup(BackupFileName, start);
            count1 = 0;
        }
        if(count2++ == MAXTRIALS){
            count2 = 0;
            cout <<
Maximum nuber of trials for this size reached.\nWe Restore the first found matrix of this size :\
;
            RestoreMatrix(MatrixSaved);
            PrintMatrix(Matrix);

            cout << "Now we try to reduce it to a critical set. ";
            long i, j, index;
            NoOfRCells = 0;
            for(i=0; i<r; i++){
                for(j=0; j<r; j++){
                    if(!Matrix[i][j].IsUnKnown()){
                        RCells[NoOfRCells++] = &Matrix[i][j];
                    }
                }
            }
            while(NoOfRCells != 0){
                index = lrand48() % NoOfRCells;

                // that can be deleted (and reconstructed again).
                DeletionTry(RCells[index]);

// Try to delete the cell
                RCells[index] = RCells[--NoOfRCells];

// Remove this cell from the list
            }
            cout << "It is :\n";
            PrintMatrix(Matrix);
            start = 0;
            for(i=0; i<r; i++){
                for(j=0; j<r; j++){
                    if(!Matrix[i][j].IsUnKnown()){
                        start++;
                    }
                }
            }
        }
    }
}

```



```

    }
    }
    CopyMatrix(MatrixSaved);
    savedstart = start;
    start++;
    cout << "So we start again by randomly selecting " << start << ...
" elements.\n";
}
}
cout << "Failed to find a critical set of size " << start << ", after " 570
<< MAXTRIALS << " trials.\nTherefore, the largest critical set is of size " ...
<< start - 1 << ".\n"; ...

return foundlarger; // true if at least one cell is deleted
}

bool RoomSquare::FindRandomSmallestCriticalSet(char *BackupFileName, int start){
    int temp, count1=0, count2=0; // Counter for backup

    // RCells are used here to point to the celles that are known
    // We only use 'start' of them 580

    if((temp = ReadBackup(BackupFileName, start)) > start){

cout << "RAND: In backup: start=" << temp << ".\nProgram uses the entered value: start=" << start < ...
    }
    else{
        start = temp;
        cout << "RAND: Program starts by: start=" << start << ".\n";
    }
    cout.flush();
    for(;;){ 590
        RandomlySelectRCells(start);
        // Randomly select RCells[0] to RCells[start-1] ...

#ifdef DEBUGESCS
        PrintMatrix(Matrix); cout.flush();
#endif
        if(CanComplete()){ // ... if the matrix can be completed.
            foundsmaller = true; // At least a CS of this size found ...

cout << "A matrix of size " << start << " is found that can uniquely complete RS.\n";
            PrintMatrix(Matrix);
            start--; // Try to find an smaller one 600
            count1 = 0;
            count2 = 0;
            system("date");
            cout << "Now looking for size " << start << " ...\n";
            cout.flush();
        }
        if(count1++ == MAXBACKUP){
            MakeBackup(BackupFileName, start);
            count1 = 0; 610
        }
        if(count2++ == MAXTRIALS){
            break;
        }
    }

cout << "Failed to find a critical set of size " << start << ", after " << MAXTRIALS << " trials.\n ...
    return foundsmaller; // true if at least one cell is deleted
}

bool RoomSquare::MoveRCellsForward(int start){ 620
    bool done = false;
    int current = start - 1;

    while(!MoveRCellFromHere(current, start)){
        if(current-- == 0){
            done = true; // All possibilities checked.
            break;
        }
    }
    return done; 630
}

```

```

bool RoomSquare::MoveRCellFromHere(int current, int start){
    bool moved = true;
    int row, col, pos, currentpos, num;

    RCells[current]->GetRowCol(row, col);
    pos = row * r + col + 1;          // NEXT POSITION
    currentpos = (pos/r + 1)*r;        // Last position of current line
    num = (r-1)/2;                    // Number of Empty positions in each row
    while((pos < r*r) && (current < start)){
        row = pos / r;
        col = pos % r;
        if(pos < currentpos){ // In the starting line
            RCells[current++] = &Matrix[row][col];
        }
        else if( ((pos % r) == 0) && ((start - current) < (r - num))){
            while(current < start){
                RCells[current++] = &Matrix[row][col++];
            }
        }
        else if(!Matrix[row][col].IsEmpty()){
            RCells[current++] = &Matrix[row][col];
        }
        pos++;
    }
    if(current < start){
        moved = false;
    }
#ifdef DEBUGRCELLMOVE
    cout << "RCells 1 to " << start << " are now: ";
    for(i=0; i<start; i++){
        RCells[i]->GetRowCol(row, col);
        cout << "(" << row+1 << ", " << col+1 << ")";
    }
    cout << '\n';
    cout.flush();
#endif

/*
    pos = row * r + col;
    if((r*r - pos) > (start - current)){
        for(i=current; i<start; i++){
            pos++;
            row = pos / r;
            col = pos % r;
            RCells[i] = &Matrix[row][col];
        }
        moved = true;
    }
*/
    return moved;
}

void RoomSquare::InitializeRCells(int start){
    int i, j, count = 0, lastfullline = start / ((r+1)/2);

    for(i=0; i<r && count<start; i++)
        for(j=0; j<r && count<start; j++)
            if(i < lastfullline){
                if( !Matrix[i][j].IsEmpty() )
                    RCells[count++] = &Matrix[i][j];
            }
            else
                RCells[count++] = &Matrix[i][j];
}

void RoomSquare::RandomlySelectRCells(int start){ // Randomly set start elements
    int i, j, row, col, *selection, num=r*r, rnd;
    bool ok1, ok2;

    selection = new int[num];
    if(!selection){
        (void) Matrix[0][0].Error("Cannot allocate memory");
    }
    for(i=0; i<r; i++){
        for(j=0; j<r; j++){
            selection[i*r+j] = i*r+j;
        }
    }
}

```

```

    SetAllUnKnown(Matrix);
    for(i=0; i<start; i++){
        do{
            if(num == 0){
                (void) Matrix[0][0].Error("Cannot select the specified number of random elements");
            }
            rnd = lrand48() % num;
            row = selection[rnd] / r;
            col = selection[rnd] % r;
            ok1 = false;
            for(j=0; j<r; j++){
                if(j != row && Matrix[j][col].IsUnKnown()){
                    ok1 = true;
                    break;
                }
            }
            ok2 = false;
            for(j=0; j<r; j++){
                if(j != col && Matrix[row][j].IsUnKnown()){
                    ok2 = true;
                    break;
                }
            }
            selection[rnd] = selection[--num];
        }while(!(ok1 && ok2));
        RCells[i] = &Matrix[row][col];
    }
    SetKnownElements(start);
    delete [] selection;
    cout << "After selecting start elements\n";
    PrintMatrix(Matrix);
}

void RoomSquare::AddAnotherRandomCell(int start){
    int i, j, row, col, e1, e2, *selection, num=r*r, rnd;
    bool ok1, ok2;

    selection = new int[num];
    if(!selection){
        (void) Matrix[0][0].Error("Cannot allocate memory");
    }
    for(i=0; i<r; i++){
        for(j=0; j<r; j++){
            selection[i*r+j] = i*r+j;
        }
    }
    for(j=0; j<start; j++){
        RCells[j]->GetRowCol(row, col);
        selection[row*r+col] = -1;
    }

    // Now remove -1 ones
    num = 0;
    for(i=0; i<r*r; i++){
        if(selection[i] != -1){
            selection[num++] = selection[i];
        }
    }

    do{
        if(num == 0){
            (void) Matrix[0][0].Error("Cannot select the specified number of random elements");
        }
        rnd = lrand48() % num;
        row = selection[rnd] / r;
        col = selection[rnd] % r;
        ok1 = false;
        for(j=0; j<r; j++){
            if(j != row && Matrix[j][col].IsUnKnown()){
                ok1 = true;
                break;
            }
        }
        ok2 = false;
        for(j=0; j<r; j++){
            if(j != col && Matrix[row][j].IsUnKnown()){

```

```

        ok2 = true;
        break;
    }
}
selection[rnd] = selection[--num];
}while(!(ok1 && ok2));

RCells[start] = &Matrix[row][col];

delete [] selection;

// Set this cell
if(MatrixOrig[row][col].GetValue(e1, e2))
    RCells[start]->SetValue(e1, e2); // Is value cell
else
    RCells[start]->SetEmpty(); // Is empty cell
cout << "After adding one element\n";
PrintMatrix(Matrix);
}
void RoomSquare::SetKnownElements(int start){
    int i, row, col, e1, e2;

    for(i=0; i<start; i++){
        RCells[i]->GetRowCol(row, col);
        if(MatrixOrig[row][col].GetValue(e1, e2))
            RCells[i]->SetValue(e1, e2); // Is value cell
        else
            RCells[i]->SetEmpty(); // Is empty cell
    }
}

void RoomSquare::SetAllUnKnown(Cell **Matrix){
    int i, j;

    for(i=0; i<r; i++)
        for(j=0; j<r; j++)
            Matrix[i][j].SetUnKnown();
}

bool RoomSquare::CheckAccuracy(){
    bool ok = true;

    if( !(r % 2) ){
        (void) Matrix[0][0].Error("Size of the room square (r) should be odd");
        ok = false;
    }

    // matrix elements are not checked!!

    if(ok)
        cout << "All conditions are checked now.\n";
    return ok;
}

bool RoomSquare::Init(){
    int i, j;

    initialized = true;
    NoOfRCells = r * r;
    RCells = (Cell **) new char[NoOfRCells * sizeof(Cell *)];
    Matrix = (Cell **) new char[r * sizeof(Cell *)];
    MatrixTemp = (Cell **) new char[r * sizeof(Cell *)];
    MatrixOrig = (Cell **) new char[r * sizeof(Cell *)];
    MatrixSaved = (Cell **) new char[r * sizeof(Cell *)];
    KnownNumbers = new int[r+1];
    RemainNumbers = new int[r+1];

    if(!Matrix || !MatrixTemp || !MatrixOrig || !MatrixSaved || !RCells || !KnownNumbers || !RemainNumbers)
        initialized = false;

    for(i=0; (i < r) && initialized; i++){
        Matrix[i] = new Cell[r];
        MatrixTemp[i] = new Cell[r];
        MatrixOrig[i] = new Cell[r];
        MatrixSaved[i] = new Cell[r];
        if(Matrix[i] && MatrixTemp[i] && MatrixOrig[i] && MatrixSaved[i]){
            for(j=0; j<r; j++){

```

```

if( Matrix[i][j].SetSize(r)  && Matrix[i][j].SetRowCol(i,j) &&
    MatrixTemp[i][j].SetSize(r) && MatrixTemp[i][j].SetRowCol(i, j) &&
    MatrixOrig[i][j].SetSize(r) && MatrixOrig[i][j].SetRowCol(i, j) &&
    MatrixSaved[i][j].SetSize(r) && MatrixSaved[i][j].SetRowCol(i, j)
    ){
    RCells[i*r+j] = &Matrix[i][j];
// or Matrix[i] + j
    }
    else
        initialized = false;
    }
    else
        initialized = false;
}

if(initialized)
    cout << "\nDynamic arrays are allocated now.\n";

return initialized;
}

void RoomSquare::PrintMatrix(Cell **Matrix){
    int i, j, num=0;
    int e1, e2;
    Cell *acell;

    cout << "--== Current Matrix Situation ==--\n";
    for(i=0; i<r; i++){
        for(j=0; j<r; j++){
            acell = &(Matrix[i][j]);
            if(acell->GetValue(e1, e2)){
                cout << setw(3) << e1 << setw(3) << e2 << " ";
                num++;
            }
            else if(acell->IsEmpty()){
                cout << " -- -- ";
                num++;
            }
            else if(acell->IsUnKnown()){
                cout << " ** ** ";
            }
            else
                acell->Error("Unexpected cell value!");
        }
        cout << '\n';
    }
    cout << "The number of known pairs is " << num << '\n';
    cout.flush();
}

void RoomSquare::PrintRCells(){
    int i, ii, jj;

    cout << "Cells: ";
    for(i=0; i<NoOfRCells; i++){
        if(RCells[i]->GetValue(ii, jj))
            cout << ii << "," << jj << " ";
        else
            cout << -2 << "," << -2 << " ";
    }
    cout << "\n";
}

bool RoomSquare::DeletionTry(Cell * acell){
    bool CellDel = false;
    bool isvalue; // false means acell is empty; true means has a value
    int e1, e2, E1, E2, num;
    int row, col;

    acell->GetRowCol(row, col);
#ifdef DEBUGDEL
    cout << "Deleting cell[" << row + 1 << "," << col + 1 << "]\n";
#endif

    isvalue = acell->GetValue(e1, e2); // If is not value, will return false

```

```

    acell->SetUnknown();
    if(!isvalue){
        e1 = e2 = EMPTY;
    }
    num = NoOfApplicablePairs(acell, Matrix, E1, E2);
    if(isvalue){
        // if(num == 1 && MaxNoOfEmptyCells(acell) == (r-1)/2){
        // if(num == 1 && E1==e1 && E2==e2 && MaxNoOfEmptyCells(acell) == (r-1)/2){
        if(num == 1){ // Can remove the value position only if num == 1 and ...
            if(CanComplete()) // ... if the matrix can be completed.
            {
                deleted = true; // At least this cell is deleted
                CellDel = true; // Don't restore the previous cell value
            }
        }
    }
    else{ // was empty
        if(num == 0){ // Can remove the empty position only if num == 0
            deleted = true; // At least this cell is deleted
            CellDel = true; // Don't restore the previous cell value
        }
    }

    // If the above couldn't
    delete the cell, apply the following (more powerful) method
    if(!CellDel){
        if(CanComplete(row, col, e1, e2)){
            // matrix can be completed toward deleted cell?
            deleted = true; // At least one cell is deleted now
            CellDel = true; // Don't restore the previous value
        }
        if(!CellDel){
            if(isvalue)
                (void)acell->SetValue(e1, e2);
            // Set the values back into the cell
            else // is empty
                acell->SetEmpty();
        }
    }

#ifdef DEBUGDEL
    if(!CellDel)
        cout << "Not ";
    cout << "Deleted cell[" << row + 1 << ", " << col + 1 << "] = ";
    if(isvalue)
        cout << "(" << e1 << ", " << e2 << ")\\n";
    else
        cout << "(EMPTY)\\n";
#endif
    return CellDel;
}

int RoomSquare::NoOfApplicablePairs(Cell *acell, Cell **Matrix, int &e1, int &e2){
    int i, j, n;
    int row, col;
    int c[4]; // contains elements of two cells -- e1,e2 , e'1,e'2
    int cNum; // Number of elements in c[]
    bool exist;
    int NoOfKnownNumbers = 0;
    // Number of known elements (in KnownNumbers[] array)
    int NoOfRemainNumbers = 0;
    // Number of remaining elements (in RemainNumbers[] array)
    int NoOfPairs = 0; // Number of applicable pairs

    acell->GetRowCol(row, col); // Row and column of this cell

    // Put all existing numbers of that row and column in KnownNumbers
    // Note that the current cell is temporarily set to UNKNOWN cell
    for(i=0; i<r; i++){
        cNum = 0;
        if(Matrix[row][i].GetValue(c[0], c[1])) // Cell in column i
            cNum = 2;
        // There are two elements that may need to be added
        if(Matrix[i][col].GetValue(c[cNum], c[cNum + 1])) // Cell is row i
            cNum += 2;
        // There are two elements that may need to be added

```

```

        for(j=0; j<cNum; j++){
            exist = false;
            for(n=0; n<NoOfKnownNumbers; n++)
// Check if already exists
                if(KnownNumbers[n] == c[j]){
                    exist = true;
                    break;
                }
            if(!exist)
                KnownNumbers[NoOfKnownNumbers++] = c[j];
// Add it if not exist
        }

// Now find the remaining elements (that can be placed in that cell) -- Compliment of KnownNumbers
for(i=0; i<=r; i++){
    exist = false;
    for(n=0; n<NoOfKnownNumbers; n++) // Check if 'i'
exists in the known list
        if(KnownNumbers[n] == i){
            exist = true;
            break;
        }
    if(!exist)
        RemainNumbers[NoOfRemainNumbers++] = i; // Add it if not exist
}

// Now find all possible combinations of pairs
if(NoOfRemainNumbers > 1) // There exists at least one pair
    for(i=0; i<(NoOfRemainNumbers - 1); i++)
        for(j=i+1; j<NoOfRemainNumbers; j++)
            if
// !CellExists(RemainNumbers[i], RemainNumbers[j], Matrix)){
                NoOfPairs++;
                e1 = RemainNumbers[i];
                e2 = RemainNumbers[j];
            }

#ifdef DEBUGPAIRS
    cout << "KnownNumbers=";
    for(i=0; i<NoOfKnownNumbers; i++)
        cout << KnownNumbers[i] << ",";
    cout << "\nRemainNumbers=";
    for(i=0; i<NoOfRemainNumbers; i++)
        cout << RemainNumbers[i] << ",";
    cout << "\NoOfPairs=" << NoOfPairs << "\n";
#endif
    return NoOfPairs;
}

bool RoomSquare::CellExists(int _e1, int _e2, Cell **Matrix){
    int i, j;
    int e1, e2;
    bool exists = false;

    for(i=0; (i<r) && !exists; i++) // For all rows
        for(j=0; j<r; j++) // and all columns
            if(Matrix[i][j].GetValue(e1, e2))
// If cell has value
                if( ((e1 == _e1) && (e2 == _e2)) ||
                    ((e1 == _e2) && (e2 == _e1)) ){ // _e1 and _e2
                        exists = true;
                        break;
                    }
    return exists;
}

void RoomSquare::CopyMatrix(Cell **NewMatrix){ // Copy Matrix to another Matrix
    int i, j;
    int e1, e2;

    for(i=0; i<r; i++) // For all rows
        for(j=0; j<r; j++){ // and all columns
            NewMatrix[i][j].SetRowCol(i, j);
            if(Matrix[i][j].GetValue(e1, e2))
// If cell has value

```

```

        (void)NewMatrix[i][j].SetValue(e1, e2);
    }
    else if(Matrix[i][j].IsEmpty())
        NewMatrix[i][j].SetEmpty();
    else
        NewMatrix[i][j].SetUnKnown();
}

void RoomSquare::RestoreMatrix(Cell **NewMatrix){          // Copy a matrix to 'Matrix'
    int i, j;
    int e1, e2;

    for(i=0; i<r; i++)          // For all rows
        for(j=0; j<r; j++){      // and all columns
            if(NewMatrix[i][j].GetValue(e1, e2))
                (void)Matrix[i][j].SetValue(e1, e2);
        }
    // Copy values to Matrix
    else if(NewMatrix[i][j].IsEmpty())
        Matrix[i][j].SetEmpty();
    else
        Matrix[i][j].SetUnKnown();
}

bool RoomSquare::CompareMatrix(Cell **NewMatrix){
    // Compare a matrix with MatrixOrig
    bool equal = true;
    int i, j;
    int e1, e2, E1, E2;

    for(i=0; (i<r) && equal; i++)          // For all rows
        for(j=0; (j<r) && equal; j++){      // and all columns
#ifdef DEBUGEQUAL
            cout << "\t Elements in [" << i+1 << ", " << j+1 << "] are ";
#endif
            if(MatrixOrig[i][j].GetValue(e1, e2)){
                if(!NewMatrix[i][j].GetValue(E1, E2) ||
                    !( ((e1 == E1) && (e2 == E2)) || ((e1 == E2) && (e2 == E1)) ) )
                    equal = false;
            }
            else if(MatrixOrig[i][j].IsEmpty() && !NewMatrix[i][j].IsEmpty())
                equal = false;
#ifdef DEBUGEQUAL
            if(equal)
                cout << "the same\n";
            else
                cout << "not the same\n";
#endif
        }
    return equal;
}

void RoomSquare::Complete(){
    if(!initialized){
        cerr << "Matrix should be read first.\n";
        return;
    }
    if(!CanComplete()){
        cout << "It couldn't be completed!\n";
    }
    PrintMatrix(MatrixTemp);
}

// Try to complete the matrix
// If the default parameter 'acell' is specified, it stops when found that cell.
// So it doesn't go to find all cells.
bool RoomSquare::CanComplete(int row, int col, int E1, int E2){
    bool done = false, changed = true, isvalue;
    int i, j, e1, e2;

    CopyMatrix(MatrixTemp); // Copy the matrix to MatrixTemp, and try to complete it

    while(!done && changed){
        // While there is at least one unknown cell and the previous loop had found a cell

```



```

done = true;
changed = false;
for(i=0; i<r; i++)          // For all rows
    for(j=0; j<r; j++){      // and all columns
        if(MatrixTemp[i][j].IsUnknown()){
            if(FindCell(&MatrixTemp[i][j])){
                changed = true;
            }
        }
    }
// At least one cell is found (to loop again)
    if(i==row && j==col){
        // This cell is the one that we really wanted to find, so stop
        isvalue = MatrixTemp[i][j].GetValue(e1, e2);    // If is not value, will return false
        if( (isvalue && e1==E1 && e2==E2) || (!isvalue && E1==EMPTY && E2==EMPTY) ){
            return true;
        }
        else{
            return false;
        }
    }
    else{
        done = false;
    }
// At lease one cell is unknown (to loop again)
    }
}
if(done){
    if(!CompareMatrix(MatrixTemp)){
        // Completed matrix is not the same as original?
        done = false;
    }
}
return done;
}

bool RoomSquare::FindCell(Cell *acell){          // Find the unknown cell in MatrixTemp
    bool found = false;
    int e1, e2;

    switch(NoOfApplicablePairs(acell, MatrixTemp, e1, e2)){
        case 0:          // It should be an empty position
            acell->SetEmpty();
            found = true;
            break;
        case 1:          // It should be value if
            if(MaxNoOfEmptyCells(acell) == ((r - 1)/2)){
                acell->SetValue(e1, e2);
                found = true;
            }
            break;
    }
}
#ifdef DEBUGCOMP
    int row, col;
    acell->GetRowCol(row, col);
    if(found)
        cout << "\t    ";
    else
        cout << "\tnot ";
    cout << "found cell[" << row+1 << ", " << col+1 << "]\n";
    if(found)
        if(acell->GetValue(e1, e2))
            cout << " as (" << e1 << ", " << e2 << ")";
        else
            cout << " as (EMPTY)";
    cout << "\n";
#endif
return found;
}

int RoomSquare::MaxNoOfEmptyCells(Cell *acell){ // Find the unknown cell in MatrixTemp
    int i;
    int NoOfEmptyCellsInRow = 0;
    int NoOfEmptyCellsInCol = 0;
    int row, col;

```

```

    acell->GetRowCol(row, col);          // Row and column of this cell          1219

    // Check all cells of that row and column
    for(i=0; i<r; i++){
        if(MatrixTemp[row][i].IsEmpty())          // Cell is empty?
            NoOfEmptyCellsInRow++;
        if(MatrixTemp[i][col].IsEmpty())          // Cell is empty?
            NoOfEmptyCellsInCol++;
    }

return (NoOfEmptyCellsInCol > NoOfEmptyCellsInRow) ? NoOfEmptyCellsInCol : NoOfEmptyCellsInRow;
}

void RoomSquare::ApproxCSBounds(char *BackupFileName){
// Approximate the upper and lower bonds for the size of cryrical sets
    int LowestSize=9999999, Lseed;
    int HighestSize=0, Hseed;
    int i, j, iter=0, seed, size, count=0;
    time_t t;

    // Read the parameters from backup file (if exist)

ReadApproxCSBoundsBackup(BackupFileName, iter, LowestSize, Lseed, HighestSize, Hseed);

//
-
    // Try to find MAXITERATIONS CSs for bound approximations
    for(; iter<MAXITERATIONS; iter++){
        seed = unsigned(mrand48() + time(&t) + iter) % MAXSEED + 1;
        srand48(seed); // initialize the RNG
        DeleteAll(); // Try each element (randomly) and try to delete it

        // Find the number of known elements
        size=0;
        for(i=0; i<r; i++){ // For all rows          1250
            for(j=0; j<r; j++){ // and all columns
                if(!Matrix[i][j].IsUnKnown()){
                    size++;
                }
            }
        }

        if(size < LowestSize){
            LowestSize = size;
            Lseed = seed;          1260
        }
        if(size > HighestSize){
            HighestSize = size;
            Hseed = seed;
        }
        if(count++ == MAXBACKUP){
            if(!MakeApproxCSBoundsBackup(BackupFileName, iter, LowestSize, Lseed, HighestSize, Hseed)){
                cout << "Error in making backup!\n";
                break;          1270
            }
            count = 0;
        }

        RestoreMatrix(MatrixOrig); // Rest the Matrix to the original shape
    }

    cout << "The following are approximations on size of CSs after " << iter << "
iterations (r=" << r << "):\n";
    cout << "Lowest CS size: " << LowestSize << ", when seed=" << Lseed << endl;
    cout << "Highest CS size: " << HighestSize << ", when seed=" << Hseed << endl;
}

```

```

const bool CheckIt = true;
const bool NoCheck = false;

const int ROW = true;
const int COLUMN = false;
//const int MAXBACKUP = 5000; // For lcs and scs
const int MAXBACKUP = 100000; // For approx

const int MAXITERATIONS = 100000; // for ApproxCSBounds() function
const int MAXSEED = 99999999; // Maximum value for seed to RNG
const int MAXTRIALS = 10000; // Maximum iteration for -rlcs and -rscs

class RoomSquare{
public:
    RoomSquare();
    ~RoomSquare();
    bool ReadMatrix(char *InFileName, bool check);
    bool FindCriticalSet();
    void ApproxCSBounds(char *BackupFileName);
    bool FindSmallestCriticalSet(char *BackupFileName, int start);
    bool FindLargestCriticalSet(char *BackupFileName, int start);
    bool FindRandomSmallestCriticalSet(char *BackupFileName, int start);
    bool FindRandomLargestCriticalSet(char *BackupFileName, int start);
    void Complete();

private:
    void ReadApproxCSBoundsBackup(char *BackupFileName, int &iter, int &LowestSize,
    int &Lseed, int &HighestSize, int &Hseed);
    bool MakeApproxCSBoundsBackup(char *BackupFileName, int iter, int LowestSize, int
    Lseed, int HighestSize, int Hseed);
    void MakeBackup(char *BackupFileName, int num);
    int ReadBackup(char *BackupFileName, int num);
    void SetAllUnknown(Cell **Matrix);
    bool CellExists(int e1, int e2, Cell **Matrix);
    int NoOfApplicablePairs(Cell *acell, Cell **Matrix, int &e1, int &e2);
    bool DeletionTry(Cell *acell);
    void DeleteAll();
    bool CannotDelete(int start);
    bool CheckAccuracy();
    bool Init();
    void PrintMatrix(Cell **Matrix);
    void PrintRCells();
    void CopyMatrix(Cell **NewMatrix);
    void RestoreMatrix(Cell **NewMatrix);
    bool CompareMatrix(Cell **NewMatrix);
    bool CanComplete(int row=-1, int col=-1, int E1=0, int E2=0); // Has default
parameters
    bool FindCell(Cell *acell);
    int MaxNoOfEmptyCells(Cell *acell);
    bool MoveRCellsForward(int start);
    bool MoveRCellFromHere(int current, int start);
    void SetKnownElements(int start);
    void InitializeRCells(int start);
    void RandomlySelectRCells(int start); // Randomly set start elements
    void AddAnotherRandomCell(int start); // Also increamens start

    int r; // Size of Room Square
    Cell **Matrix; // 2x2 matrix
    Cell **MatrixSaved; // 2x2 matrix to save a useful case
    Cell **MatrixTemp; // 2x2 temporary matrix
    Cell **MatrixOrig; // 2x2 original matrix
    Cell **RCells; // Array of cells that can be deleted or ...
    int NoOfRCells; // Number of above cells
    bool initialized; // true, when Matrix is initialized
    bool deleted; // true, when at least one cell is deleted
    bool partial; // true if a partial matrix is entered
    bool foundsmaller; // true, when at least one smaller CS is found
    bool foundlarger; // true, when at least one larger CS is found
    int *KnownNumbers;

    int *RemainNumbers; // List of all numbers that don
    't exist in row and column corresponding to a cell
};

```



```

#ifndef CELLCLASS
#define CELLCLASS

enum bool {false, true};

const int VALUE = 0;    // Cell has a normal value (a pair of integers)
const int UNKNOWN = -1; // Cell has unknown value
const int EMPTY = -2;  // Cell hash empty value

class Cell{           // This class represents a cell of matrix (Room Square)
public:
    Cell();           // Constructor
    ~Cell();          // Destructor

    bool SetSize(int r);

    /*inline*/ bool SetValue(int e1, int e2);
    /*inline*/ bool GetValue(int &e1, int &e2);
    /*inline*/ bool IsValue();

    bool SetRowCol(int i, int j);
    void GetRowCol(int &i, int &j);
    int GetRow();
    int GetCol();

    /*inline*/ void SetUnKnown();
    /*inline*/ bool IsUnKnown();

    /*inline*/ void SetEmpty();
    /*inline*/ bool IsEmpty();

    bool Error(char *str);           // Prints an error message and returns false
    unsigned long NoOfChecks;        // No of times attempted to deleted this cell

private:
    int r;                          // Size of Matrix (maximum size)
    int row, col;                   // Row and column of this cell (start from zero)
    int element1, element2;         // Two elements of a cell
    int status;                    // Status of the cell (VALUE, UNKNOWN, or EMPTY)
};

#endif

```

```

#include <iostream.h>
#include "Cell.h"

Cell::Cell(){
    status = UNKNOWN;           // To be set in ReadMatrix()
    r = -1;                     // To be set in SetSize()
    NoOfChecks = 0;             // No of times attempted to deleted this cell
}

Cell::~Cell(){
    // Nothing yet
}

bool Cell::SetSize(int _r){
    if( _r >= 0 ){
        r = _r;
        return true;
    }
    else
        return Error("Size (r) is out of range");
}

bool Cell::SetRowCol(int i, int j){
    if( (i >= 0) && (i < r) && (j >= 0) && (j < r) ){
        row = i;
        col = j;
        return true;
    }
    else
        return Error("Row and/or column are out of range");
}

void Cell::GetRowCol(int &i, int &j){
    i = row;
    j = col;
}

int Cell::GetRow(){
    return row;
}

int Cell::GetCol(){
    return col;
}

bool Cell::SetValue(int e1, int e2){
    if( (e1 >= 0) && (e1 <= r) && (e2 >= 0) && (e2 <= r) ){
        element1 = e1;
        element2 = e2;
        status = VALUE;
        return true;
    }
    else
        return Error("Elements are out of range");
}

bool Cell::GetValue(int &e1, int &e2){
    if(IsValue()){
        e1 = element1;
        e2 = element2;
        return true;
    }
    else
        return false;
}

bool Cell::IsValue(){
    return bool(status == VALUE);
}

void Cell::SetUnKnown(){
    status = UNKNOWN;
}

bool Cell::IsUnKnown(){
    return bool(status == UNKNOWN);
}

```

```
void Cell::SetEmpty(){
    status = EMPTY;
}

bool Cell::IsEmpty(){
    return bool(status == EMPTY);
}

bool Cell::Error(char *str){
    cerr << "\nError - " << str << ".\n";
    return false;
}
```

79  
80

```

#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iomanip.h>

#include "LatinSquare.h"

void Usage(char *fname){
    cout << "Usage is:  " << fname << " -complete      SeedValue MatrixFileName\n";
    cout << "or      :  " << fname << " -forcecomplete SeedValue MatrixFileName\n";
    cout << "or      :  " << fname << " -cs          SeedValue MatrixFileName\n";
    cout << "or      :  " << fname << " -forcecs     SeedValue MatrixFileName\n";
    cout << "or      :  " << fname << "
- slbcs      SeedValue MatrixFileName BackupFileName\n";
    cout << "or      :  " << fname << "
- forceslbcs SeedValue MatrixFileName BackupFileName\n";
    cout << "or      :  " << fname << "
- allcs      Order      MatrixFileName [print]\n";
}

main(int argc, char *argv[]){
    char *option;
    long SeedValue;
    char *MatrixFileName, *BackupFileName;
    int StartSize;
    LatinSquare square;

    if(argc <= 3)
        Usage(argv[0]);
    else{
        option = argv[1];
        sscanf(argv[2], "%d", &SeedValue);          // Seed for
        RNG (also order of LS for -allcs)
        srand48(SeedValue);                          // Randomize the RNG
        MatrixFileName = argv[3];
        if(square.ReadMatrix(MatrixFileName, CheckIt)){
            if(strcmp(option, "-forcecomplete") == 0){
                square.ForceComplete();
            }
            else if(strcmp(option, "-complete") == 0){
                square.Complete();
            }
            else if(strcmp(option, "-cs") == 0){
                if(square.FindCriticalSet(false))
            }
            else if(strcmp(option, "-forcecs") == 0){
                if(square.FindCriticalSet(true))
            }
        }
        // Find Critical Set
        cout << "Critical Set was found.\n";
        else
            cout << "Couldn't delete any more!\n";
        }
        else if(strcmp(option, "-forceslbcs") == 0){
            if(square.FindCriticalSet(true))
        }
        // Find Critical Set using force method
        cout <<
    }
    else
        cout <<
    "Couldn't delete any more, using force method!\n";
    }
    else if(strcmp(option, "-slbcs") == 0){
        BackupFileName = argv[4];
        square.ApproxCSBounds(BackupFileName, false);
    }
    // Find Bounds for Critical Sets
    }
    else if(strcmp(option, "-forceslbcs") == 0){
        BackupFileName = argv[4];
        square.ApproxCSBounds(BackupFileName, true);
    }
    -for Critical Sets
    }
    else if(strcmp(option, "-allcs") == 0){
        bool print = false;
        if(argc == 5 && strcmp(argv[4], "print")==0){
            print = true;
        }
    }
}

```



```

    }
    int num = square.FindAllCS(SeedValue, print);
// Find all CSs of order "SeedValue"
    cout << "Number of CSs of order " << SeedValue << " is "
    << num << endl;
    }
    else
        Usage(argv[0]);
    }
}
```

66  
...  
...  
70

```

#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include <iomanip.h>
#include "LatinSquare.h"

// #define DEBUGDEL
// #define DEBUGCOMP
// #define DEBUGFIND
// #define DEBUGRCCELLMOVE
// #define DEBUGESCS

LatinSquare::LatinSquare(){
    n = 0;
    NoOfRCCells = 0;
    partial = false;
    initialized = false;
    deleted = false;
}

LatinSquare::~LatinSquare(){
    int i;

    if(initialized){
        for(i=0; i<n; i++){
            delete [] Matrix[i];
            delete [] MatrixTemp[i];
            delete [] MatrixOrig[i];
        }
        delete [] Matrix;
        delete [] MatrixTemp;
        delete [] MatrixOrig;
        delete [] RCells;
        delete [] KnownNumbers;
        delete [] RemainNumbers;
    }
}

bool LatinSquare::ReadMatrix(char *InFileName, bool check){
    int i, j;
    Cell element;
    bool ok = true;
    ifstream file(InFileName);

    NoOfRCCells = 0;
    if(file.good()){
        file >> n;
        if(file.good() && Init()){
            for(i=0; (i<n) && ok; i++){
                for(j=0; (j<n) && ok; j++){
                    file >> element;
                    if(!FromZero && element >= 0){
                        element--;
                    }
                    if(file.fail()){
                        ok = false;
                    }
                    else if( !ElementInRange(element) ){
                        Error("Element out of range");
                        ok = false;
                    }
                    else{
                        if(element == UNKNOWN){
                            partial = true;
                        }
                        else{
                            RCells[NoOfRCCells++] = &Matrix[i][j];
                            Matrix[i][j] = element;
                        }
                    }
                }
            }
        }
        for(i=0; (i<n) && ok; i++){

```

```

        for(j=0; (j<n) && ok; j++){
            if(partial){
                file >> element;
                if(!FromZero && element >= 0){
                    element--;
                }
                if(file.fail()){
                    ok = false;
                }
                else if( !ElementInRange(element) ){
                    Error("Element out of range");
                    ok = false;
                }
                else{
                    if(element == UNKNOWN){
                        Error(
                            "Second matrix should not have any unknown element");
                        ok = false;
                    }
                    else{
                        MatrixOrig[i][j] = element;
                    }
                }
            }
            else{
                MatrixOrig[i][j] = Matrix[i][j];
            }
        }
    }
    else
        ok = false;
    file.close();
}
else ok = false;

if(ok)
    ok = check ? CheckAccuracy() : true;
else
    Error("Cannot work with input file");

#ifdef DEBUGREAD
    cout << "Test started\n";
    PrintMatrix(Matrix);
    PrintRCells();
    cout << "Test finished\n";
#endif

if(ok)
    cout << "Reading the matrix is successfully completed.\n\n";

cout.flush();

CopyMatrix(MatrixOrig);

return ok;
}

bool LatinSquare::FindCriticalSet(bool force){
    if(!initialized){
        return Error("Matrix should be read first");
    }

    DeleteAll(force);
    PrintMatrix(Matrix);
    return deleted; // true if at least one cell is deleted
}

void LatinSquare::DeleteAll(bool force){
    long i, j, index;

    NoOfRCells = 0;
    for(i=0; i<n; i++){
        for(j=0; j<n; j++){
            RCells[NoOfRCells++] = &Matrix[i][j];    // or Matrix[i] + j
        }
    }
}

```

```

    while(NoOfRCells){
        index = lrand48() % NoOfRCells;

// Remove the unwanted cells. They are either necessary cells or the ones
// that can be deleted (and reconstructed again).
        (void)DeletionTry(*RCells[index], force);

        RCells[index] = RCells[--NoOfRCells];
// Remove this cell from the list
#ifdef DEBUGFIND
        PrintRCells();
#endif
    }
}

bool LatinSquare::CheckAccuracy(){
    bool ok = true;

    if(n < 2){
        ok = Error("Size of the latin square (n) should be at least 2");
    }

    // matrix elements are not checked!!

    if(ok)
        cout << "All conditions are checked now.\n";
    return ok;
}

bool LatinSquare::Init(){
    int i, j;

    initialized = true;
    int MaxNoOfRCells = n * n;
    RCells = (Cell **) new char[MaxNoOfRCells * sizeof(Cell *)];
    Matrix = (Cell **) new char[n * sizeof(Cell *)];
    MatrixTemp = (Cell **) new char[n * sizeof(Cell *)];
    MatrixOrig = (Cell **) new char[n * sizeof(Cell *)];
    KnownNumbers = new int[n];
    RemainNumbers = new int[n];
    if
(!Matrix || !MatrixTemp || !MatrixOrig || !RCells || !KnownNumbers || !RemainNumbers){
        initialized = false;
    }

    for(i=0; (i < n) && initialized; i++){
        Matrix[i] = new Cell[n];
        MatrixTemp[i] = new Cell[n];
        MatrixOrig[i] = new Cell[n];
        if(!Matrix[i] || !MatrixTemp[i] || !MatrixOrig[i]){
            initialized = false;
        }
    }

    if(initialized)
        cout << "\nDynamic arrays are allocated now.\n";

    return initialized;
}

void LatinSquare::PrintMatrix(Cell **NewMatrix){
    int i, j, num=0;
    Cell acell;

    cout << "---= Current Matrix Situation =--\n";
    for(i=0; i<n; i++){
        for(j=0; j<n; j++){
            acell = NewMatrix[i][j];
            if(acell != UNKNOWN){
                if(!FromZero){
                    acell++;
                }
                cout << setw(3) << acell << " ";
                num++;
            }
            else{

```

```

        cout << " ** ";
    }
    cout << '\n';
}
cout << "The number of known elements is " << num << '\n';
cout.flush();
}

void LatinSquare::PrintRCells(){
    int i, ii, jj;

    cout << "Cells: ";
    for(i=0; i<NoOfRCells; i++){
        if(*(RCells[i]) != UNKNOWN)
            cout << *(RCells[i]) << " ";
        else
            cout << "*** ";
    }
    cout << endl;
}

bool LatinSquare::DeletionTry(Cell &acell, bool force){
    Cell element;

    bool can;
    element = acell;
    acell = UNKNOWN; // Set it to an unknown position (temporarily)

    can = force ? CanForceComplete() : CanComplete(true);
    if(can && CompareMatrix(MatrixTemp)){
        // If the matrix can be completed and the result is the same as original, ...
        return(deleted = true);
        // then at least this cell is deleted successfully
    }

    acell = element;
    return false; // not deleted
}

int LatinSquare::NoOfApplicableElements(Cell **NewMatrix, int row, int
col, Cell &acell){
    int i, j, k;
    int c[2]; // contains elements of two cells -- e1,e2 , e'1,e'2
    int cNum; // Number of elements in c[]
    bool exist;
    int NoOfKnownNumbers = 0;
    // Number of known elements (in KnownNumbers[] array)
    int NoOfRemainNumbers = 0;
    // Number of remaining elements (in RemainNumbers[] array)
    int NoOfElements = 0; // Number of applicable elements

    // Put all existing numbers of that row and column in KnownNumbers
    // Note that the current cell is temporarily set to UNKNOWN cell
    for(i=0; i<n; i++){
        cNum = 0;
        if( (c[0]=NewMatrix[row][i]) != UNKNOWN ) // Cell in column i
            cNum++;

        if( (c[cNum]=NewMatrix[i][col]) != UNKNOWN ) // Cell is row i
            cNum++;

        // There is one element that may need to be added
        for(j=0; j<cNum; j++){ // Add c[] to the list (if
any) -- At most 4 elements
            exist = false;
            for(k=0; k<NoOfKnownNumbers; k++) // Check if
already exists
                if(KnownNumbers[k] == c[j]){
                    exist = true;
                    break;
                }
            if(!exist)
                KnownNumbers[NoOfKnownNumbers++] = c[j];
        }
    }
    // Add it if not exist
}

```

```

// Now find the remaining elements (that can be placed in that cell) -- Compliment of KnownNumbers set
for(i=0; i<n; i++){
    exist = false;
    for(k=0; k<NoOfKnownNumbers; k++) // Check if 'i
// exists in the known list
        if(KnownNumbers[k] == i){
            exist = true;
            break;
        }
    if(!exist)
        RemainNumbers[NoOfRemainNumbers++] = i; // Add it if not exist
}

NoOfElements = NoOfRemainNumbers;
acell = RemainNumbers[0];

#ifdef DEBUGPAIRS
    cout << "KnownNumbers=";
    for(i=0; i<NoOfKnownNumbers; i++)
        cout << KnownNumbers[i] << ", ";
    cout << "\nRemainNumbers=";
    for(i=0; i<NoOfRemainNumbers; i++)
        cout << RemainNumbers[i] << ", ";
    cout << "\nNoOfElements=" << NoOfElements << "\n";
#endif
return NoOfElements;
}

void LatinSquare::CopyMatrix(Cell **NewMatrix){ // Copy Matrix to another Matrix
    int i, j;

    for(i=0; i<n; i++){ // For all rows
        for(j=0; j<n; j++){ // and all columns
            NewMatrix[i][j] = Matrix[i][j];
        }
    }
}

void LatinSquare::RestoreMatrix(){ // Copy MatrixOrig to Matrix
    int i, j;

    for(i=0; i<n; i++){ // For all rows
        for(j=0; j<n; j++){ // and all columns
            Matrix[i][j] = MatrixOrig[i][j];
        }
    }
}

bool LatinSquare::CompareMatrix(Cell **NewMatrix){
// Compare a matrix with MatrixOrig
    bool equal = true;
    int i, j;

    for(i=0; (i<n) && equal; i++) // For all rows
        for(j=0; (j<n) && equal; j++){ // and all columns
#ifdef DEBUGEQUAL
            cout << "\t Elements in [" << i+1 << ", " << j+1 << "] are ";
            if(MatrixOrig[i][j] != NewMatrix[i][j])
                equal = false;
#endif
            if(equal)
                cout << "the same\n";
            else
                cout << "not the same\n";
        }
    }
    return equal;
}

bool LatinSquare::CanComplete(bool copyTTemp){ // Try to complete the matrix
    bool done = false, changed = true;
    int i, j;

    if(copyTTemp){
        CopyMatrix(MatrixTemp);
// Copy the matrix to MatrixTemp, and try to complete it
    }
}

```

```

    while(!done && changed){
// While there is at least one unknown cell and the previous loop had found a cell
    done = true;
    changed = false;
    for(i=0; i<n; i++)          // For all rows
        for(j=0; j<n; j++){    // and all columns
            if(MatrixTemp[i][j] == UNKNOWN){
                if(FindCell(MatrixTemp, i, j, false))
                    changed = true;
            }
        }
// At least one cell is found (to loop again)
    else
        done = false;
}

    }

}

return done;
}

bool LatinSquare::FindCell(Cell **NewMatrix, int row, int col, bool force){
// Find the unknown cell in MatrixTemp
    bool found = false, ok;
    Cell element;
    int i, j, e, NoOfElements;

    if( (NoOfElements = NoOfApplicableElements(NewMatrix, row, col, element)) == 1){
        NewMatrix[row][col] = element;
        found = true;
    }
    else if(force){
        for(e=0; e<NoOfElements; e++){
            element = RemainNumbers[e];

// Check whether element cannot be in any other UNKNOWN element of same row
            ok = true;
            for(j=0; j<n; j++){
                if(NewMatrix[row][j]==UNKNOWN && j!=col && !CellExistCol(NewMatrix, element, j)){
                    ok = false;
                    break;
                }
            }
            if(!ok){
                ok = true;
                for(i=0; i<n; i++){
                    if(NewMatrix[i][col]==UNKNOWN && i!=row && !CellExistRow(NewMatrix, element, i)){
                        ok = false;
                        break;
                    }
                }
            }
            if(ok){
                NewMatrix[row][col] = element;
                found = true;
                break;
            }
        }
    }

}

#ifdef DEBUGCOMP
    if(found)
        cout << "\t\t ";
    else
        cout << "\t\t not ";
    cout << "found cell[" << row+1 << ", " << col+1 << "]\n";
    if(found)
        cout << " as (" << int(element)+1 << ")\n";
    cout << "\n";
#endif
return found;
}

```

371

...

...

380

...

390

...

...

410

...

420

430

```

bool LatinSquare::CellExistRow(Cell **NewMatrix, Cell element, int row){
    int j;
    for(j=0; j<n; j++){
        if(NewMatrix[row][j] == element){
            return true;
        }
    }
    return false;
}

bool LatinSquare::CellExistCol(Cell **NewMatrix, Cell element, int col){
    int i;
    for(i=0; i<n; i++){
        if(NewMatrix[i][col] == element){
            return true;
        }
    }
    return false;
}

bool LatinSquare::ElementInRange(Cell acell){
    return ((acell == UNKNOWN || (acell>=0 && acell<=n)) ? true : false);
}

void LatinSquare::Complete(){
    if(!initialized){
        Error("Matrix should be read first");
        return;
    }
    if(!CanComplete(true)){
        cout << "It couldn't be completed!\n";
    }
    PrintMatrix(MatrixTemp);
}

void LatinSquare::ForceComplete(){
    if(!initialized){
        Error("Matrix should be read first");
        return;
    }
    if(!CanForceComplete()){
        cout << "It couldn't be completed by forcing method!\n";
    }
    PrintMatrix(MatrixTemp);
}

bool LatinSquare::CanForceComplete(){
    bool done, changed, copy = true;;
    int i, j;

    // First try normal complete, then force method (and repeat)
    do{
        done = true;
        changed = false;
        if(!CanComplete(copy)){
            copy = false;
            done = false;
            for(i=0; i<n && !changed; i++){ // For all rows
                for(j=0; j<n && !changed; j++){ // and all columns
                    if(MatrixTemp[i][j] == UNKNOWN && FindCell(MatrixTemp, i, j, true)){
                        changed = true;
                    }
                }
            }
        }
    }while(!done && changed);
    return done;
}

bool LatinSquare::Error(char *str){
    cerr << "\nError - " << str << ".\n";
    return false;
}

```



517

```
bool LatinSquare::MakeApproxCSBoundsBackup(char *BackupFileName, int iter, int LowestSize, int Ls
bool ok = true;
ofstream file;
char tmpstr[1000];
```

520

```
sprintf(tmpstr, "mv %s %s.bak", BackupFileName, BackupFileName);
system(tmpstr);
file.open(BackupFileName, ios::out | ios::trunc);
if(file.good()){
```

```
    file    << iter << endl
           << LowestSize << ' ' << Lseed << endl
           << HighestSize << ' ' << Hseed << endl
```

530

```
<< "Lowest CS size: " << LowestSize << ", when seed=" << Lseed << endl
```

```
<< "Highest CS size: " << HighestSize << ", when seed=" << Hseed << endl;
```

```
    if(file.fail()){
        cerr << "Error -- Cannot write to backup file.\n";
        ok = false;
    }
    file.close();
```

540

```
    else{
        cerr << "Error -- Cannot create backup file.\n";
        ok = false;
    }
```

```
    return ok;
}
```

```
void LatinSquare::ReadApproxCSBoundsBackup(char *BackupFileName, int &iter, int &LowestSize, int &
ifstream file;
```

```
file.open(BackupFileName, ios::in);
if(file.good()){
    file >> iter >> LowestSize >> Lseed >> HighestSize >> Hseed;
    if(file.fail()){
        cerr << "Cannot read from backup file.\n";
        exit(1);
    }
    file.close();
}
```

560

```
// Otherwise, this is the first time, and there is no backup!
```

```
void LatinSquare::ApproxCSBounds(char *BackupFileName, bool force){
// Approximate the upper and lower bonds for the size of cryrical sets
int LowestSize=9999999, Lseed;
int HighestSize=0, Hseed;
int i, j, iter=0, seed, size, count=0;
time_t t;
```

570

```
// Read the parameters from backup file (if exist)
```

```
// cout << BackupFileName << ' ' << iter << ' ' << LowestSize << ' ' << Lseed << '
' << HighestSize << ' ' << Hseed << endl;
```

...

```
// Try to find MAXITERATIONS CSs for bound approximations
for(; iter<MAXITERATIONS; iter++){
    seed = unsigned(mrand48() + time(&t) + iter) % MAXSEED + 1;
    srand48(seed); // initialize the RNG
    DeleteAll(force);
```

580

```
// Try each element (randomly) and try to delete it

// Find the number of known elements
size=0;
for(i=0; i<n; i++){ // For all rows
    for(j=0; j<n; j++){ // and all columns
        if(Matrix[i][j] != UNKNOWN){
```

```

        size++;
    }
}

if(size < LowestSize){
    LowestSize = size;
    Lseed = seed;
}
if(size > HighestSize){
    HighestSize = size;
    Hseed = seed;
}
if(count++ == MAXBACKUP){
    if(!MakeApproxCSBoundsBackup(BackupFileName, iter, LowestSize, Lseed, HighestSize, Hseed)){
        cout << "Error in making backup!\n";
        break;
    }
    count = 0;
}

RestoreMatrix();// Rest the Matrix to the original shape

cout << "The following are approximations on size of CSs after " << iter << " iterations (n=" << n << "
cout << "Lowest CS size: " << LowestSize << ", when seed=" << Lseed << endl;
cout << "Highest CS size: " << HighestSize << ", when seed=" << Hseed << endl;

}

int LatinSquare::FindAllCS(int order, int print){ // Find all CSs of order "order"

    bool done = false;
    int i, j, count, ind; // Counter for backup
    int *RCellRows = new int[n*n];
    int *RCellCols = new int[n*n];
    if(RCellRows == NULL || RCellCols == NULL){
        cerr << "Cannot allocate memory!\n";
        return 0;
    }

    // RCells are used here to point to the celles that are known
    // We only use 'order' of them

    // Initialize RCells[] to the first start elements
    ind = 0;
    for(i=0; i<n && ind<order; i++){
        for(j=0; j<n && ind<order; j++){
            RCells[ind] = &Matrix[i][j];
            RCellRows[ind] = i;
            RCellCols[ind] = j;
            ind++;
        }
    }

    // Go to all possible "order" elements and check whether it can be completed
    count = 0;
    do{
        // Set all elements to unknown
        for(i=0; i<n; i++){
            for(j=0; j<n; j++){
                Matrix[i][j] = UNKNOWN;
            }
        }

        // Based of RCells[0] to RCells[order-1] set "order elements
        for(i=0; i<order; i++){
            *RCells[i] = MatrixOrig[RCellRows[i]][RCellCols[i]];
        }

        if(CanComplete(true)){ // ... if the matrix can be completed.
            if(print){
                PrintMatrix(Matrix);
            }
            count++;
        }
    } while(!done);
}

#ifdef DEBUGSCS

```

```

        else(
            cout << "The following cannot be completed! (count="
" << count << ")\n";
            PrintMatrix(Matrix);
        )
#endif
        cout.flush();
    }while(!MoveRCellsForward(order, RCells, RCellRows, RCellCols));
// Next possibility (exhaustive search)

    delete RCellRows;
    delete RCellCols;
    return count;    // number of LS with "order" elements that can be completed
}

bool LatinSquare::MoveRCellsForward(int order, Cell **RCells, int *RCellRows, int *RCellCols){
    bool done = false;
    int current = order - 1;

    while(!MoveRCellFromHere(current, order, RCells, RCellRows, RCellCols)){
        if(current-- == 0){
            done = true;    // All possibilities checked.
            break;
        }
    }
    return done;
}

bool LatinSquare::MoveRCellFromHere(int current, int order, Cell **RCells, int *RCellRows, int *R
    bool moved = true;
    int i, row, col, pos, currentpos, num;

    row = RCellRows[current];
    col = RCellCols[current];
    pos = row * n + col + 1;    // NEXT POSITION
    currentpos = (pos/n + 1)*n;    // Last position of current line
    while((pos < n*n) && (current < order)){
        row = pos / n;
        col = pos % n;
        if(pos < currentpos){    // In the starting line
            RCells[current] = &Matrix[row][col];
            RCellRows[current] = row;
            RCellCols[current] = col;
            current++;
        }
        else if( ((pos % n) == 0) && ((order - current) < n)){
            while(current < order){
                RCells[current] = &Matrix[row][col];
                RCellRows[current] = row;
                RCellCols[current] = col;
                current++;
                col++;
            }
        }
        pos++;
    }
    if(current < order){
        moved = false;
    }

#ifdef DEBUGRCELLMOVE
    if(moved){
        cout << "RCells 1 to " << order << " are now : ";
        for(i=0; i<order; i++){
            cout << "(" << RCellRows[i]+1 << ", " << RCellCols[i]+1 << ") ";
        }
        cout << '\n';
        cout.flush();
    }
#endif
    return moved;
}

```



```

#ifndef __LATINSQUARE__
#define __LATINSQUARE__

#include "MyTypes.h"

typedef int Cell;          // cell value range: 0-n, and -1 means unknown
const int UNKNOWN = -1; // Cell has unknown value

const bool CheckIt = true;
const bool NoCheck = false;

const int ROW = true;
const int COLUMN = false;
const int MAXBACKUP = 500;

const int MAXITERATIONS = 100000; // for ApproxCSBounds() function
const int MAXSEED = 99999999; // Maximum value for seed to RNG

const bool FromZero = false; // false: 0, 1, 2, ..., n-1
                             // true: 1, 2, 3, ..., n

class LatinSquare{

public:
    LatinSquare();
    ~LatinSquare();
    bool ReadMatrix(char *InFileName, bool check);
    bool FindCriticalSet(bool force);
    void Complete(); // To complete a partial LS
    void ForceComplete(); // To complete a partial LS by force method
    void ApproxCSBounds(char *BackupFileName, bool force);
    int FindAllCS(int order, int print); // Find all CSs of order "order"

private:
    bool MoveRCellsForward(int order, Cell **RCells, int *RCellRows, int
    *RCellCols);
    bool MoveRCellFromHere(int current, int order, Cell **RCells, int *RCellRows,
    int *RCellCols);

    void ReadApproxCSBoundsBackup(char *BackupFileName, int &iter, int &LowestSize,
    - &Lseed, int &HighestSize, int &Hseed);
    bool MakeApproxCSBoundsBackup(char *BackupFileName, int iter, int LowestSize,
    int Lseed, int HighestSize, int Hseed);

    int NoOfApplicableElements(Cell **Matrix, int row, int col, Cell &acell);
    bool DeletionTry(Cell &acell, bool force);
    void DeleteAll(bool force);
    bool CheckAccuracy();
    bool Init();
    void PrintMatrix(Cell **Matrix);
    void PrintRCells();
    void CopyMatrix(Cell **NewMatrix);
    void RestoreMatrix();
    bool CompareMatrix(Cell **NewMatrix);
    bool CanComplete(bool copyToTemp);
    bool CanForceComplete();
    bool FindCell(Cell **NewMatrix, int row, int col, bool force);
    bool ElementInRange(Cell acell);
    bool Error(char *str); // Prints an error message and returns false
    bool CellExistRow(Cell **NewMatrix, Cell element, int row);
    bool CellExistCol(Cell **NewMatrix, Cell element, int col);

    int n; // Size of Latin Square
    Cell **Matrix; // nxn matrix
    Cell **MatrixTemp; // nxn temporary matrix
    Cell **MatrixOrig; // nxn original matrix
    Cell **RCells; // Array of cells that can be deleted or ...
    int NoOfRCells; // Number of above cells
    bool initialized; // true, when Matrix is initialized
    bool deleted; // true, when at least one cell is deleted
    bool partial; // true, when original matrix is partial
    bool foundsmaller; // true, when at least one smaller CS is found
    bool foundlarger; // true, when at least one larger CS is found
    int *KnownNumbers;

    // List of all numbers that exist in row and column corresponding to a cell

```

```
int *RemainNumbers;    // List of all numbers that don
";
};
#endif
```

Feb 18 13:36 1997

MyTypes.h 1

```
typedef unsigned char bool;
typedef unsigned char binary;
typedef unsigned char BYTE;

typedef unsigned int UINT;

typedef unsigned long ULONG;

const bool false = 0;
const bool true = 1;
```

10

```

#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include <iomanip.h>
#include "Bhaskar.h"

// #define DEBUG

main(int argc, char *argv[]){
    Bhaskar bhaskar;
    int num;

    if(argc <= 2)
        cout << "Usage is:  " << argv[0] << " InputFileName BackupFileName\n";
    else{
        if(bhaskar.ReadMatrix(argv[1], argv[2], CheckIt)){
            // Read matrix (CheckIt or NoCheck)
            num = bhaskar.Find(OnlyOne);
            cout << num << " Bhaskars were found.\n";
        }
    }

    Bhaskar::Bhaskar(){
        v = b = r = k = y = 0;
        row = 0;
        initialized = false;
    }

    Bhaskar::~Bhaskar(){
        if(initialized){
            delete [] RowSituation;
            for(i=0; i<v; i++){
                delete [] Matrix[i];
                delete [] NonZeroCol[i];
                for(j=0; j<v; j++)
                    delete [] CommonCol[i][j];
                delete [] CommonCol[i];
            }
            delete [] Matrix;
            delete [] NonZeroCol;
            delete [] CommonCol;
        }
    }

    bool Bhaskar::ReadMatrix(char *InFileName, char *BakFileName, bool check){
        int i, j, n, h, element;
        bool ok=true;
        ifstream file;

        BackupFileName = BakFileName;

        file.open(InFileName, ios::in);
        if(file.good()){
            file >> v >> b >> r >> k >> y >> firstrow;
            firstrow--;
            if(file.good() && Init()){
                for(i=0; i<v; i++){
                    for(j=0; j<b; j++){
                        file >> element;
                        Matrix[i][j] = -element;
                        if(file.fail()){
                            ok = false;
                            break;
                        }
                    }
                }
            }
            else ok = false;
            file.close();
        }
        else ok = false;

        if(ok)
            ok = check ? CheckAccuracy() : true;
        else

```



```

        cerr << "Cannot work with input file.\n";
    if(ok){
        for(i=0; i<v; i++){
            n = 0;
            for(j=0; j<b; j++){
                if(abs(Matrix[i][j]) == 1)
                    NonZeroCol[i][n++] = j;
            }
            for(h=0; h<v; h++){
                if(h != i){
                    n = 0;
                    for(j=0; j<b; j++){
                        if
                            CommonCol[i][h][n++] = j;
                    }
                }
            }
        }
    }

#ifdef DEBUG
    cout << "Test started\n";
    PrintFoundBhaskar();
    cout << "-----\n";
    for(i=0; i<v; i++){
        for(j=0; j<r; j++){
            cout << NonZeroCol[i][j] << " ";
        }
        cout << "\n";
    }
    cout << "-----\n";
    for(i=0; i<v; i++){
        for(h=0; h<v; h++){
            if(h != i){
                cout << "Rows " << i << " and " << h << " : ";
                for(j=0; j<y; j++){
                    cout << CommonCol[i][h][j] << " ";
                }
                cout << "\n";
            }
        }
    }
    cout << "-----\n";
    for(i=0; i<v; i++){
        cout << "Row situation for row " << i << " is " << RowSituation[i] <<
        "\n";
    }
    cout << "Test finished\n";
#endif

    row = firstrow;
    if(ok)
        ok = ReadBackup();

    if(ok)
        cout << "Reading the matrix is successfully completed.\n\n";
    cout.flush();
    return ok;
}

int Bhaskar::Find(bool one){
    bool done = false, found = false;
    int count = 0;
    int printedrow = 3;
    int loops = 0;

    if(!initialized){
        cerr << "Matrix should be read first.\n";
        done = true;
    }
    else
        cout << "Now, starting the search ...\n\n";

    while(!done){
        if( RowSituation[row] == RowPossibilities ){
            // The last possible situation in this row.
            if(row == firstrow){
                cout << "All possibilities checked.\n";
                done = true;
            }
            else{
                RowSituation[row] = 0;
                // Set situation in this row to zero.
            }
        }
    }
}

```

```

RowSituation[--row]++;
...
    }
    else{
        SetSituation();
        if(InnerProduct()){
            // Go to the next row with this situation in this row.
            if(++row > printedrow){
                int i,j;
                cout << "Partial solution when row " << row <<
...
                for(i=0; i<row; i++){
                    for(j=0; j<b; j++)
...
                cout << setw(3) << -Matrix[i][j];
                cout << "\n";
            }
            cout <<
...
            "-----\n";
            cout.flush();
            printedrow = row;
...
        }
        if(row == v){
            PrintFoundBhaskar();
            RowSituation[--row]++;
            count++;
            if(one)
                done = true;
        }
        else{ // Inner product is not zero.
            RowSituation[row]++;
...
// Go to next situation in the same row
        }

        // Backup if needed
        if( ++loops == SavingPeriod ){
            if(!Backup()){
                cerr << "Cannot make a backup!\n";
                break;
            }
            loops = 0;
...
        }
    }
    return count;
}

bool Bhaskar::CheckAccuracy(){
    bool ok = true;
    int i, j, h, n;

    if( firstrow < 0 || firstrow >= v ){
        cerr << "firsrtrow should be a positive number less than v.\n";
        ok = false;
    }
    else if( (v*r) != (b*k) ){
        cerr << "vr = " << v*r << " but bk = " << b*k << "\n";
        ok = false;
    }
    else if( (y*(v-1)) != (r*(k-1)) ){
        cerr << "y(v-1) = " << y*(v-1) << " but r(k-1) = " << r*(k-1) << "\n";
        ok = false;
    }
    else if( (r < 2) || (r > 31) ){
        cerr << "r should be between 2 and 31.\n";
        ok = false;
    }
    else{
        for(i=0; (i<v) && ok; i++){
            n = 0;
            for(j=0; j<b; j++){
                if(abs(Matrix[i][j]) == 1)
                    n++;
                else if(Matrix[i][j] != 0){
                    cerr << "The cell in row " << i <<
...
                    " and column " << j << " is not valid.\n";

```

```

                                ok = false;
                                break;
                                }
    if(n != r){
        cerr << n << "There is not " << r << " times 1 in row "
        ok = false;
        break;
    }
    for(h=0; h<v; h++)
        if(h != i){
            n = 0;
            for(j=0; j<b; j++)
                if
((abs(Matrix[i][j]) == 1) && (abs(Matrix[h][j]) == 1))
                    n++;
            if(n != y){
                cerr << "There is not " << y <<
... << i << " and " << h << "\n";
                ok = false;
                break;
            }
        }
    for(j=0; (j<b) && ok; j++){
        n = 0;
        for(i=0; i<v; i++)
            if(abs(Matrix[i][j]) == 1)
                n++;
        if(n != k){
            cerr << "There is not " << k << " times 1 in column "
            ok = false;
            break;
        }
        for(h=0; h<b; h++)
            if(h != j){
                n = 0;
                for(i=0; i<v; i++)
                    if
                        n++;
                if(n != y){
                    cerr << "There is not " << y <<
" same ones in columns " << j << " and " << h << "\n";
                    ok = false;
                    break;
                }
            }
        }
    }
    if(ok)
        cout << "All conditions are checked now.\n";
    return ok;
}

void Bhaskar::SetSituation(){
    int i;
    int rr = r - 2;
    int RS = RowSituation[row];

    for(i=0; i<rr; i++)
        Matrix[ row ][ NonZeroCol[row][i] ] = ((RS >> (rr-i)) & TWO) - 1;
        Matrix[ row ][ NonZeroCol[row][r-2] ] = (RS & TWO) - 1;
        Matrix[ row ][ NonZeroCol[row][r-1] ] = ((RS << 1) & TWO) - 1;
}

bool Bhaskar::Init(){
    initialized = true;
    int i, j;

    RowPossibilities = ONE << (r-1);
    RowSituation = new int[v];
    if(RowSituation){
        for(i=0; i<v; i++){
            RowSituation[i] = 0;
        }
    }
}

```

```

else{
    cerr << "Cannot allocate memory.\n";
    initialized = false;
}
if(initialized){
    Matrix = (int **) new char[v * sizeof(int *)];
    if(!Matrix)
        initialized = false;
    for(i=0; (i < v) && initialized; i++){
        Matrix[i] = new int[b];
        if(!Matrix[i])
            initialized = false;
    }
}
if(initialized){
    NonZeroCol = (int **) new char[v * sizeof(int *)];
    if(!NonZeroCol)
        initialized = false;
    for(i=0; (i < v) && initialized; i++){
        NonZeroCol[i] = new int[r];
        if(!NonZeroCol[i])
            initialized = false;
    }
}
if(initialized){
    CommonCol = (int ***) new char[v * sizeof(int *)];
    if(!CommonCol)
        initialized = false;
    for(i=0; (i < v) && initialized; i++){
        CommonCol[i] = (int **) new char[v * sizeof(int *)];
        if(!CommonCol[i])
            initialized = false;
        else{
            for(j=0; (j < v) && initialized; j++){
                CommonCol[i][j] = new int[y];
                if(!CommonCol[i][j])
                    initialized = false;
            }
        }
    }
}
if(initialized)
    cout << "\nDynamic arrays are allocated now.\n";

return initialized;
}

bool Bhaskar::InnerProduct(){
    int i, j, c, result;
    bool ok = true;

    for(i=0; i<row; i++){
        result = 0;
        for(j=0; j<y; j++){
            c = CommonCol[row][i][j];
            result += Matrix[row][c] * Matrix[i][c];
            result += (Matrix[row][c] ^ Matrix[i][c]) | ONE;
        }
        if(result != 0){ // Inner product is not zero
            ok = false;
            break;
        }
    }
    return ok;
}

void Bhaskar::PrintFoundBhaskar(){
    int i, j;

    cout << "--== Found one Bhaskar ==--\n";
    for(i=0; i<v; i++){
        for(j=0; j<b; j++)
            cout << setw(3) << -Matrix[i][j];
        cout << "\n";
    }
}

bool Bhaskar::Backup(){

```

```

bool ok = true;
int i, j;
ofstream file;
char tmpstr[1000];

sprintf(tmpstr, "mv %s %s.bak", BackupFileName, BackupFileName);
system(tmpstr);

file.open(BackupFileName, ios::out | ios::trunc);
if(file.good()){
    file << row << "\n\n";
    if(file.good()){
        for(i=0; i<v; i++){
            file << RowSituation[i] << " ";
            if(file.fail()){
                ok = false;
                break;
            }
        }
        file << "\n\n";
        for(i=0; ok && (i<v); i++){
            for(j=0; j<b ; j++){
                file << setw(3) << -Matrix[i][j];
                if(file.fail()){
                    ok = false;
                    break;
                }
            }
            file << "\n";
            if(file.fail()){
                ok = false;
                break;
            }
        }
    }
    else{
        ok = false;
    }
    file.close();
}
else
    ok = false;

return ok;
)

bool Bhaskar::ReadBackup(){
    bool ok = true;
    int i, j;
    ifstream file;
    int element;

    file.open(BackupFileName, ios::in);
    if(file.good()){
        cout << "Backup file (";
        file >> row;
        cout << row << ": ";
        if(file.good()){
            for(i=0; i<v; i++){
                file >> RowSituation[i];
                cout << RowSituation[i] << ", ";
                if(file.fail()){
                    ok = false;
                    break;
                }
            }
            for(i=0; ok && (i<v); i++){
                for(j=0; j<b ; j++){
                    file >> element;
                    Matrix[i][j] = -element;
                    if(file.fail()){
                        ok = false;
                        break;
                    }
                }
            }
        }
        else
            ok = false;
    }
}

```

```
        file.close();
        cout << ")\n";
    }
    // Otherwise, this is the first time, and there is no backup!
    if(!ok)
        cerr << "Error in reading " << BackupFileName << " file.\n";
    return ok;
}
```

```
enum bool {false, true};

const bool OnlyOne = true;
const bool All      = false;

const bool CheckIt = true;
const bool NoCheck = false;

const int ONE = 0x00000001;
const int TWO = 0x00000002;
10

const int SavingPeriod = 100000000; // Number of loops before making backup

class Bhaskar{
private:

    int v, b, r, k, y; // y is Lambda
    int row, firstrow;
    char *BackupFileName;
    int RowPossibilities;
    int *RowSituation;
    int **NonZeroCol;
    int ***CommonCol; //
    int **Matrix; // 2x2 matrix
    bool initialized; // true, when Matrix is initialized

    bool CheckAccuracy();
    bool Init();
    inline void SetSituation();
    inline bool InnerProduct();
    void PrintFoundBhaskar();
    bool Backup();
    bool ReadBackup();
    20
public:

    Bhaskar();
    ~Bhaskar();
    bool ReadMatrix(char *, char *, bool);
    int Find(bool);
    30
};
40
```

```

#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include <iomanip.h>
#include "Bhaskar64.h"

// #define DEBUG

main(int argc, char *argv[]){
    Bhaskar bhaskar;
    int num;

    if(argc <= 2)
        cout << "Usage is:  " << argv[0] << " InputFileName BackupFileName\n";
    else{
        if(bhaskar.ReadMatrix(argv[1], argv[2], CheckIt)){
            // Read matrix (CheckIt or NoCheck)
            num = bhaskar.Find(OnlyOne); // Find (OnlyOne or All)
            cout << num << " Bhaskars were found.\n";
        }
    }
}

Bhaskar::Bhaskar(){
    v = b = r = k = y = 0;
    row = 0;
    initialized = false;
}

Bhaskar::~Bhaskar(){
    if(initialized){
        delete [] RowSituation;
        for(i=0; i<v; i++){
            delete [] Matrix[i];
            delete [] NonZeroCol[i];
            for(j=0; j<b; j++){
                delete [] CommonCol[i][j];
                delete [] CommonCol[i];
            }
            delete [] Matrix;
            delete [] NonZeroCol;
            delete [] CommonCol;
        }
    }
}

bool Bhaskar::ReadMatrix(char *InFileName, char *BakFileName, bool check){
    int i, j, n, h, element;
    bool ok=true;
    ifstream file;

    BackupFileName = BakFileName;

    file.open(InFileName, ios::in);
    if(file.good()){
        file >> v >> b >> r >> k >> y >> firstrow;
        firstrow--;
        if(file.good() && Init()){
            for(i=0; i<v; i++){
                for(j=0; j<b; j++){
                    file >> element;
                    Matrix[i][j] = -element;
                    if(file.fail()){
                        ok = false;
                        break;
                    }
                }
            }
        }
        else ok = false;
        file.close();
    }
    else ok = false;

    if(ok)
        ok = check ? CheckAccuracy() : true;
    else

```



```

    cerr << "Cannot work with input file.\n";
    if(ok){
        for(i=0; i<v; i++){
            n = 0;
            for(j=0; j<b; j++){
                if(abs(Matrix[i][j]) == 1)
                    NonZeroCol[i][n++] = j;
            }
            for(h=0; h<v; h++){
                if(h != i){
                    n = 0;
                    for(j=0; j<b; j++){
                        if
                            CommonCol[i][h][n++] = j;
                    }
                }
            }
        }
    }

#ifdef DEBUG
    cout << "Test started\n";
    PrintFoundBhaskar();
    cout << "-----\n";
    for(i=0; i<v; i++){
        for(j=0; j<r; j++){
            cout << NonZeroCol[i][j] << " ";
        }
        cout << "\n";
    }
    cout << "-----\n";
    for(i=0; i<v; i++){
        for(h=0; h<v; h++){
            if(h != i){
                cout << "Rows " << i << " and " << h << " : ";
                for(j=0; j<y; j++){
                    cout << CommonCol[i][h][j] << " ";
                }
                cout << "\n";
            }
        }
    }
    cout << "-----\n";
    for(i=0; i<v; i++){
        cout << "Row situation for row " << i << " is " << RowSituation[i] <<
        "\n";
    }
    cout << "Test finished\n";
#endif

    row = firstrow;
    if(ok)
        ok = ReadBackup();

    if(ok)
        cout << "Reading the matrix is successfully completed.\n\n";
    cout.flush();
    return ok;
}

int Bhaskar::Find(bool one){
    bool done = false, found = false;
    int count = 0;
    int printedrow = 3;
    int loops = 0;

    if(!initialized){
        cerr << "Matrix should be read first.\n";
        done = true;
    }
    else
        cout << "Now, starting the search ...\n\n";

    while(!done){
        if( RowSituation[row] == RowPossibilities ){
            // The last possible situation in this row.
            if(row == firstrow){
                cout << "All possibilities checked.\n";
                done = true;
            }
            else{
                RowSituation[row] = 0;
                // Set situation in this row to zero.
            }
        }
    }
}

```

78

80

90

100

110

...

140

150

...

```

RowSituation[--row]++;
...
    }
    else{
        SetSituation();
        if(InnerProduct()){
            // Go to the next row with this situation in this row.
            if(++row > printedrow){
                int i,j;
                cout << "Partial solution when row " << row <<
...
                for(i=0; i<row; i++){
                    for(j=0; j<b; j++)
...
            }
            cout << "\n";
            cout <<
...
            cout.flush();
            printedrow = row;
170
        }
        if(row == v){
            PrintFoundBhaskar();
            RowSituation[--row]++;
            count++;
            if(one)
                done = true;
        }
        else{ // Inner product is not zero.
            RowSituation[row]++;
180
            // Go to next situation in the same row
        }

        // Backup if needed
        if( ++loops == SavingPeriod ){
            if(!Backup()){
                cerr << "Cannot make a backup!\n";
                break;
            }
            loops = 0;
190
        }
    }
}
return count;
}

bool Bhaskar::CheckAccuracy(){
    bool ok = true;
    int i, j, h, n;

    if( firstrow < 0 || firstrow >= v ){
        cerr << "fisrtrow should be a positive number less than v.\n";
        ok = false;
    }
    else if( (v*r) != (b*k) ){
        cerr << "vr = " << v*r << " but bk = " << b*k << "\n";
        ok = false;
    }
    else if( (y*(v-1)) != (r*(k-1)) ){
        cerr << "y(v-1) = " << y*(v-1) << " but r(k-1) = " << r*(k-1) << "\n";
210
        ok = false;
    }
    else if((r < 2) || (r > 62)){
        cerr << "r should be between 2 and 62.\n";
        ok = false;
    }
    else{
        for(i=0; (i<v) && ok; i++){
            n = 0;
            for(j=0; j<b; j++){
220
                if(abs(Matrix[i][j]) == 1)
                    n++;
                else if(Matrix[i][j] != 0){
                    cerr << "The cell in row " << i <<
...
                    " and column " << j << " is not valid.\n";

```

```

                                ok = false;
                                break;
                                }
                                if(n != r){
<< i << "\n";
                                cerr << n << "There is not " << r << " times 1 in row "
                                ok = false;
                                break;
                                }
                                for(h=0; h<v; h++){
                                    if(h != i){
                                        n = 0;
                                        for(j=0; j<b; j++){
                                            if
((abs(Matrix[i][j]) == 1) && (abs(Matrix[h][j]) == 1))
                                                n++;
                                            if(n != y){
                                                cerr << "There is not " << y <<
... << i << " and " << h << "\n";
                                                ok = false;
                                                break;
                                            }
                                        }
                                    }
                                for(j=0; (j<b) && ok; j++){
                                    n = 0;
                                    for(i=0; i<v; i++){
                                        if(abs(Matrix[i][j]) == 1)
                                            n++;
                                    }
                                    if(n != k){
                                        cerr << "There is not " << k << " times 1 in column "
<< j << "\n";
                                        ok = false;
                                        break;
                                    }
                                for(h=0; h<b; h++){
                                    if(h != j){
                                        n = 0;
                                        for(i=0; i<v; i++){
                                            if
                                                n++;
                                            if(n != y){
                                                cerr << "There is not " << y <<
" same ones in columns " << j << " and " << h << "\n";
                                                ok = false;
                                                break;
                                            }
                                        }
                                    }
                                }
                                if(ok)
                                    cout << "All conditions are checked now.\n";
                                return ok;
}

void Bhaskar::SetSituation(){
    int i;
    int rr = r - 2;
    LLONG RS = RowSituation[row];

    for(i=0; i<rr; i++)
        Matrix[ row ][ NonZeroCol[row][i] ] = (int(RS >> (rr-i)) & TWO) - 1;
    Matrix[ row ][ NonZeroCol[row][r-2] ] = (int(RS) & TWO) - 1;
    Matrix[ row ][ NonZeroCol[row][r-1] ] = ((int(RS) << 1) & TWO) - 1;
}

bool Bhaskar::Init(){
    initialized = true;
    int i, j;

    RowPossibilities = LLONG(ONE) << (r-1); // 2^r
    RowSituation = new LLONG[v];
    if(RowSituation){
        for(i=0; i<v; i++){
            RowSituation[i] = 0;
        }
    }
}

```

```

else{
    cerr << "Cannot allocate memory.\n";
    initialized = false;
}
if(initialized){
    Matrix = (int **) new char[v * sizeof(int *)];
    if(!Matrix)
        initialized = false;
    for(i=0; (i < v) && initialized; i++){
        Matrix[i] = new int[b];
        if(!Matrix[i])
            initialized = false;
    }
}
if(initialized){
    NonZeroCol = (int **) new char[v * sizeof(int *)];
    if(!NonZeroCol)
        initialized = false;
    for(i=0; (i < v) && initialized; i++){
        NonZeroCol[i] = new int[r];
        if(!NonZeroCol[i])
            initialized = false;
    }
}
if(initialized){
    CommonCol = (int ***) new char[v * sizeof(int *)];
    if(!CommonCol)
        initialized = false;
    for(i=0; (i < v) && initialized; i++){
        CommonCol[i] = (int **) new char[v * sizeof(int *)];
        if(!CommonCol[i])
            initialized = false;
        else{
            for(j=0; (j < v) && initialized; j++){
                CommonCol[i][j] = new int[y];
                if(!CommonCol[i][j])
                    initialized = false;
            }
        }
    }
}
if(initialized)
    cout << "\nDynamic arrays are allocated now.\n";

return initialized;
}

bool Bhaskar::InnerProduct(){
    int i, j, c, result;
    bool ok = true;

    for(i=0; i<row; i++){
        result = 0;
        for(j=0; j<y; j++){
            c = CommonCol[row][i][j];
            result += Matrix[row][c] * Matrix[i][c];
            result += (Matrix[row][c] ^ Matrix[i][c]) | ONE;
        }
        if(result != 0){ // Inner product is not zero
            ok = false;
            break;
        }
    }
    return ok;
}

void Bhaskar::PrintFoundBhaskar(){
    int i, j;

    cout << "--- Found one Bhaskar ---\n";
    for(i=0; i<v; i++){
        for(j=0; j<b; j++){
            cout << setw(3) << -Matrix[i][j];
        }
        cout << "\n";
    }
}

bool Bhaskar::Backup(){

```

```

bool ok = true;
int i, j;
ofstream file;
char tmpstr[1000];

sprintf(tmpstr, "mv %s %s.bak", BackupFileName, BackupFileName);
system(tmpstr);

file.open(BackupFileName, ios::out | ios::trunc);
if(file.good()){
    file << row << "\n\n";
    if(file.good()){
        for(i=0; i<v; i++){
            file << RowSituation[i] << " ";
            if(file.fail()){
                ok = false;
                break;
            }
        }
        file << "\n\n";
        for(i=0; ok && (i<v); i++){
            for(j=0; j<b ; j++){
                file << setw(3) << -Matrix[i][j];
                if(file.fail()){
                    ok = false;
                    break;
                }
            }
            file << "\n";
            if(file.fail()){
                ok = false;
                break;
            }
        }
    }
    else{
        ok = false;
    }
    file.close();
}
else
    ok = false;

return ok;
}

bool Bhaskar::ReadBackup(){
    bool ok = true;
    int i, j;
    ifstream file;
    int element;

    file.open(BackupFileName, ios::in);
    if(file.good()){
        cout << "Backup file (";
        file >> row;
        cout << row << ": ";
        if(file.good()){
            for(i=0; i<v; i++){
                file >> RowSituation[i];
                cout << RowSituation[i] << ", ";
                if(file.fail()){
                    ok = false;
                    break;
                }
            }
            for(i=0; ok && (i<v); i++){
                for(j=0; j<b ; j++){
                    file >> element;
                    Matrix[i][j] = -element;
                    if(file.fail()){
                        ok = false;
                        break;
                    }
                }
            }
        }
    }
    else
        ok = false;
}

```

```
        file.close();
        cout << ")\n";
    }
    // Otherwise, this is the first time, and there is no backup!
    if(!ok)
        cerr << "Error in reading " << BackupFileName << " file.\n";
    return ok;
}
```

```

typedef long long LLONG;

const bool OnlyOne = true;
const bool All      = false;

const bool CheckIt = true;
const bool NoCheck = false;

const int ONE = 0x00000001;
const int TWO = 0x00000002;
10

const int SavingPeriod = 100000000;    // Number of loops before making backup

class Bhaskar{
private:
    int v, b, r, k, y;    // y is Lambda
    int row, firstrow;
    char *BackupFileName;
    LLONG RowPossibilities;
    LLONG *RowSituation;
    int **NonZeroCol;
    int ***CommonCol;    //
    int **Matrix;        // 2x2 matrix
    bool initialized;    // true, when Matrix is initialized

    bool CheckAccuracy();
    bool Init();
    inline void SetSituation();
    inline bool InnerProduct();
    void PrintFoundBhaskar();
    bool Backup();
    bool ReadBackup();
    20
public:
    Bhaskar();
    ~Bhaskar();
    bool ReadMatrix(char *, char *, bool);
    int Find(bool);
    30
};
40

```

Feb 18 13:36 1997

MyTypes.h 1

```
typedef unsigned char bool;  
typedef unsigned char binary;  
typedef unsigned char BYTE;
```

```
typedef unsigned int UINT;
```

```
typedef unsigned long ULONG;
```

```
const bool false = 0;  
const bool true = 1;
```

10



```

/*****
brd.c
    Reads in a file describing a group of sets and tests whether they are
    SDS. If so attempts to find BRD for the sets.
    Fabian Magrini - 9590218
    CSCI965 - Project - Autumn 1997

@input file format
<integer=m> <integer=v> <integer=k> <integer=lamda>
m lines of k*<integers>

where:
    m is the number of sets
    v is the modulus of the sets
    k is the number of elements in the sets

*****/

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

/*****
    Defines
*****/
#define NONBAR 0
#define BAR 1

/*****
    Types
*****/

/*-----
    Error codes
-----*/
typedef enum {
    NO_ERROR,
    FAILED_INPUT_FILE_OPEN,
    FAILED_INPUT_FILE_READ,
    UNEXPECTED_END_OF_FILE,
    FAILED_OUTPUT_FILE_OPEN,
    OUT_OF_MEMORY,
    INVALID_REQUEST_ALLOCATE_MEMORY,
    INVALID_PARAMETER,
    DIVIDE_BY_ZERO
} ERROR_CODE;

typedef struct set_item_struct {
    int value;
    int sign;
} set_item;

typedef struct result_item_struct {
    int count;
    int bar_count;
} result_item;

/*****
    Global Variables
*****/

int gM, gV, gK, gLamda;
set_item **gSets;
int *gSetsTrial;
result_item *gResults;
int gMaxNumOfSetTypes;
int gTotalNumOfEach;
int *gSetTypes;
int *gSetTypesCount;
int gInitTrial;

/*-----
    Error messages
-----*/
char *gErrorMessages[]={
    "No error",
    "Failed to open input file",

```

```

    "Failed to read input file",
    "Unexpected end of file",
    "Failed to open output file",
    "Out of memory",
    "Invalid request to allocate memory",
    "Invalid parameter passed to function",
    "Divide by zero"
};

```

79  
80

```

/*****

```

### Forward Declarations

```

*****/

```

90

```

void Error(ERROR_CODE code);
void ReadInputFile(char *name);
void Dump(void);
void DumpSets(void);
void DumpResults(void);
void DumpSetsTrial(void);
void Initialise(void);
void Finalise(void);
set_item **CreateSets(int size);
set_item *CreateSet(int size);
set_item *ReadSet(FILE *file, int size);
void PrintSet(set_item set[], int size);
void CreateResults(void);
void InitialiseResults(void);
int CheckSDS(void);
int FindConstraints(void);
void CreateSetsTrial(void);
void InitialiseSetsTrial(void);
int NextSetsTrial(void);
void InitialiseBarMix(int item);
int NextBarMix(int item, int bars);
int CheckBRD(void);
int FindBRD(void);

```

100

110

```

/*****

```

### Main

```

*****/

```

```

void main(int argc, char *argv[]) {
    printf("Begin ...\n");
    if (argc < 2) {
        printf("Usage: brd <input file>\n");
    } else {
        Initialise();
        ReadInputFile(argv[1]);
        if (CheckSDS()) {
            DumpResults();
            DumpSets();
            FindConstraints();
            if (FindBRD()) {
                DumpResults();
            }
            DumpSets();
        } else {
            DumpResults();
            DumpSets();
        }
        Finalise();
    }
}

```

120

130

140

```

/*****

```

### Function Definitions

```

*****/

```

```

/*-----
    Error:
    Prints the error message.
    @param code - error code
    -----*/

```

```

void Error(ERROR_CODE code) {
    fprintf(stderr, "**** ERROR: %s.\n", gErrorMessages[code]);
}

```

150

```

/*-----

```

### ReadInputFile:

Read the input file and store the data into global structures.

@param name - name of the input file  
 @error&exit FAILED\_INPUT\_FILE\_OPEN  
 @error&exit FAILED\_INPUT\_FILE\_READ  
 @error&exit DIVIDE\_BY\_ZERO  
 @comment  
 attributes read and stored in the following globals gM, gV, gK.  
 sets read and stored in global gSets dynamically created as an array  
 of gM pointers to an array of gK set\_item.  
 dynamically creates gResults as an array of gV result\_item ;

```

-----*/
void ReadInputFile(char *name) {
    FILE *file;
    int i;

    if ((name == NULL) || ((file=fopen(name, "r"))==NULL)) {
        Error(FAILED_INPUT_FILE_OPEN);
        exit(FAILED_INPUT_FILE_OPEN);
    }

    if (fscanf(file, "%d%d%d", &gM, &gV, &gK, &gLamda)) {
        if (gLamda == 0) {
            if ((gV-1)==0) {
                Error(DIVIDE_BY_ZERO);
                exit(DIVIDE_BY_ZERO);
            } else gLamda = (2*gK*(gK-1))/(gV-1);
        }
        fprintf(stdout, "gM=%d gV=%d gK=%d\n", gM, gV, gK);

        gSets = CreateSets(gM);

        for (i=0; i<gM; i++) {
            fprintf(stdout, "gSets[%d]=", i);
            gSets[i] = ReadSet(file, gK);
        }

    } else {
        Error(FAILED_INPUT_FILE_READ);
        exit(FAILED_INPUT_FILE_READ);
    }

    fclose(file);
    CreateResults();
}

/*-----
CreateSets:
Allocate memory for sets array.
@param size - number of sets
@return pointer to sets array
@error&exit OUT_OF_MEMORY
@error&exit INVALID_REQUEST_ALLOCATE_MEMORY
-----*/
set_item **CreateSets(int size) {
    set_item **set;
    int i;

    set = NULL;
    if (size > 0) {
        set=(set_item **)malloc(size*sizeof(set_item *));
        if (set==NULL) {
            Error(OUT_OF_MEMORY);
            exit(OUT_OF_MEMORY);
        }
        for (i=0; i<size; i++) {
            set[i] = NULL;
        }
    } else {
        Error(INVALID_REQUEST_ALLOCATE_MEMORY);
        exit(INVALID_REQUEST_ALLOCATE_MEMORY);
    }
    return (set);
}

/*-----
CreateSet:
Allocate memory for a set.
@param size - size of set

```

```

235  @return pointer to set
236  @error&exit OUT_OF_MEMORY
237  @error&exit INVALID_REQUEST_ALLOCATE_MEMORY
238  -----*/
239  set_item *CreateSet(int size) {
240      set_item *set;
241      int i;
242
243      set = NULL;
244      if (size > 0) {
245          set=(set_item *)malloc(size*sizeof(set_item));
246          if (set==NULL) {
247              Error(OUT_OF_MEMORY);
248              exit(OUT_OF_MEMORY);
249          }
250          for (i=0; i<size; i++) {
251              set[i].value = 0;
252              set[i].sign = 0;
253          }
254      } else {
255          Error(INVALID_REQUEST_ALLOCATE_MEMORY);
256          exit(INVALID_REQUEST_ALLOCATE_MEMORY);
257      }
258      return (set);
259  }
260  -----*/
261
262  ReadSet:
263  Create a set of "size" and read the set values from the input file.
264  @param file - file to read set from
265  @param size - size of set
266  @return pointer to set
267  @error&exit FAILED_INPUT_FILE_OPEN
268  -----*/
269  set_item *ReadSet(FILE *file, int size){
270      set_item *set;
271      int i;
272
273      set = CreateSet(size);
274
275      for (i=0; i<size; i++) {
276          if (fscanf(file, "%d", &set[i].value))
277              fprintf(stdout, "%d ", set[i].value);
278          else {
279              Error(FAILED_INPUT_FILE_READ);
280              exit(FAILED_INPUT_FILE_READ);
281          }
282      }
283      fprintf(stdout, "\n");
284      return (set);
285  }
286  -----*/
287
288  PrintSet:
289  Output the contents of a set to stdout.
290  @param set - an array of "size" "set_item"s
291  @param size - size of set
292  -----*/
293  void PrintSet(set_item set[], int size) {
294      int i;
295      for (i=0; i<size; i++) {
296          if (set[i].sign == BAR)
297              fprintf(stdout, "-%d ", set[i].value);
298          else
299              fprintf(stdout, "%d ", set[i].value);
300      }
301      fprintf(stdout, "\n");
302  }
303  -----*/
304
305  Initialise:
306  Initialise global data structures.
307  -----*/
308  void Initialise(void) {
309      gM=0;
310      gV=0;
311      gK=0;
312      gLamda=0;

```

313

```

gSets=NULL;
gMaxNumOfSetTypes=0;
gSetTypes=NULL;

```

}

/\*-----

Finialise:

Deconstruct global structures, returning any allocated memory.

320

/\*-----\*/

void Finalise(void) {

int i;

if (gSets != NULL) {

for (i=0;i&lt;gM;i++) {

free(gSets[i]);

}

free(gSets);

}

if (gResults != NULL) free(gResults);

330

if (gSetTypes != NULL) free(gSetTypes);

}

/\*-----

Dump:

Output the contents of global data structures.

/\*-----\*/

void Dump(void) {

fprintf(stdout, "gM=%d gV=%d gK=%d gLamda=%d\n", gM, gV, gK, gLamda);

340

}

/\*-----

DumpSets:

Output the contents of global sets structure.

/\*-----\*/

void DumpSets(void) {

int i;

fprintf(stdout, "gSets:\n");

for (i=0;i&lt;gM;i++) {

350

fprintf(stdout, "gSets[%d]=" , i);

PrintSet(gSets[i], gK);

}

}

/\*-----

DumpResults:

Output the contents of global results data structures.

/\*-----\*/

void DumpResults(void) {

360

int i;

fprintf(stdout, "gResults:\n");

for (i=0;i&lt;gV;i++)

fprintf(stdout, "%d %d\n", i, gResults[i].count);

}

/\*-----

DumpSetsTrial:

Output the contents of gSetsTrial.

/\*-----\*/

void DumpSetsTrial(void) {

370

int i;

fprintf(stderr, "gSetsTrial: ");

for (i=0;i&lt;gM;i++) {

fprintf(stderr, "%d ", gSetsTrial[i]);

}

fprintf(stderr, "\n");

}

/\*-----

CreateResults:

Allocate memory for results.

@assume gV - the size of the results array

@error&amp;exit OUT\_OF\_MEMORY

@error&amp;exit INVALID\_REQUEST\_ALLOCATE\_MEMORY

/\*-----\*/

void CreateResults(void) {

380

if (gV &gt; 0) {

gResults=(result\_item \*)malloc(gV\*sizeof(result\_item));

390

if (gResults==NULL) {

```

        Error(OUT_OF_MEMORY);
        exit(OUT_OF_MEMORY);
    }
} else {
    Error(INVALID_REQUEST_ALLOCATE_MEMORY);
    exit(INVALID_REQUEST_ALLOCATE_MEMORY);
}
InitialiseResults();
}

/*-----
    InitialiseResults:
    Reset gResults to initial state.
    @assume gResults
    @assume gV - the size of the results array
    -----*/
void InitialiseResults(void) {
    int i;
    for (i=0; i<gV; i++) {
        gResults[i].count = 0;
        gResults[i].bar_count = 0;
    }
}

/*-----
    CheckSDS:
    Check and report whether gSets are supplementary difference sets.
    Sets v, b, r, k, l are supplementary difference sets (v-1)l=2k(k-1) if all
    the non-zero differences occur the same number of times l.
    @return int which is true if SDS otherwise false
    @assume gSets
    @assume gResults - has been initialised
    @assume gK - the number of items in set
    @assume gM - the number of sets in gSets
    @comment
    Find all the differences between elements of gSets[m] for m = 0 to (gM-1).
    And store in gResults[d]. Where d is the differene value. If gResults
    contains each non-zero element of gV a fixed number of times (lamda) then
    the sets are supplementary difference sets m-{v,k,lamda}. Set gLamda.
    -----*/
int CheckSDS(void) {
    int i,j,m,result;
    set_item *set;
    int occur;

    /* Calculate results */
    for (i=0; i<gK; i++) {
        for (j=0; j<gK; j++) {
            for (m=0; m<gM; m++) {
                set = gSets[m];
                result = set[j].value - set[i].value;
                if (result < 0) result += gV;
                gResults[result].count++;
            }
        }
    }

    /* Check results - consider only non-zero results*/
    occur = 0;
    for (i=1; i<gV; i++) {
        if (gResults[i].count>0) {
            if (occur == 0) occur = gResults[i].count;
            else if (gResults[i].count != occur) break;
        }
    }

    if (i==gV) {
        fprintf(stdout, "SDS\n");
        gLamda = occur;
        return 1;
    } else {
        fprintf(stdout, "Not SDS\n");
        return 0;
    }
}

/*-----
    FindConstraints:
    Determines the constraints for finding BRD from the SDS.
    @return int which is true if constraints found otherwise false
    -----*/

```

```

@assume gSets
@assume gK - the number of items in sets
@assume gM - the number of sets in gSets
@assume gLamda - the needed results occurrence
@comment
Calculates and stores gMaxNumOfSetTypes, the maximum number of
set types.
Dynamically creates gSetTypes as an array of gMaxNumOfSetTypes
integers. Calculates and stores the solution to equations (1)-(3)
given in theory section of paper on project, the combination of
set types that may give BRDs
Dynamically creates gSetTypesCount as an array of gMaxNumOfSetTypes
integers.
-----*/
int FindConstraints(void) {
    int *PosSetTypes;
    int *NegSetTypes;
    int i;
    int Total, PosTotal;

    Dump();

    if ((gK % 2) == 0)
        gMaxNumOfSetTypes = (gK / 2);
    else
        gMaxNumOfSetTypes = ((gK + 1) / 2);
    fprintf(stdout, "gMaxNumOfSetTypes=%d\n", gMaxNumOfSetTypes);

    PosSetTypes=(int *)malloc(gMaxNumOfSetTypes*sizeof(int));
    NegSetTypes=(int *)malloc(gMaxNumOfSetTypes*sizeof(int));
    gSetTypes=(int *)malloc(gMaxNumOfSetTypes*sizeof(int));
    gSetTypesCount=(int *)malloc(gMaxNumOfSetTypes*sizeof(int));

    for (i=0;i<gMaxNumOfSetTypes;i++) {
        NegSetTypes[i] = 2 * i * (gK - i);
        PosSetTypes[i] = (i * (i - 1)) + ((gK - i) * ((gK - i) - 1));
    }

    fprintf(stdout, "PosSetTypes:\n");
    for (i=0;i<gMaxNumOfSetTypes;i++) {
        fprintf(stdout, "PosSetTypes[%d]=%d\n", i, PosSetTypes[i]);
    }
    fprintf(stdout, "NegSetTypes:\n");
    for (i=0;i<gMaxNumOfSetTypes;i++) {
        fprintf(stdout, "NegSetTypes[%d]=%d\n", i, NegSetTypes[i]);
    }

    gTotalNumOfEach = ((gV - 1) * gLamda) / 2 ;
    fprintf(stdout, "gTotalNumOfEach=%d\n", gTotalNumOfEach);

    Total=0;
    for (i=0;i<gMaxNumOfSetTypes;i++) gSetTypes[i]=0;
    while (Total != gM) {
        i=gMaxNumOfSetTypes-1;
        gSetTypes[i]++;
        while (gSetTypes[i]>gM) {
            gSetTypes[i]=0;
            i--;
            if (i<0) break;
            gSetTypes[i]++;
        }
        if (i<0) break;
        Total=0;
        PosTotal=0;
        for (i=0;i<gMaxNumOfSetTypes;i++) {
            Total += gSetTypes[i];
            PosTotal += gSetTypes[i] * PosSetTypes[i];
            fprintf(stdout, "(%d, %d) ", gSetTypes[i], PosSetTypes[i]);
        }
        fprintf(stdout, "= (%d, %d)\n", Total, PosTotal);
        if (PosTotal!=gTotalNumOfEach) Total=0;
    }
    free(PosSetTypes);
    free(NegSetTypes);

    if (i<0) {
        fprintf(stdout, "Constraints not found!\n");
    }
}

```

469  
470

480

490

500

510

520

530

540

```

        return 0;
    } else {
        fprintf(stdout, "Constraints found.\n");
        return 1;
    }
}

/*-----
   CreateSetsTrial:
   Allocate memory for trial.
   @assume gM - the number of sets in gSets
   @error&exit OUT_OF_MEMORY
   @error&exit INVALID_REQUEST_ALLOCATE_MEMORY
   -----*/
void CreateSetsTrial(void) {
    if (gM > 0) {
        gSetsTrial=(int *)malloc(gM*sizeof(int));
        if (gSetsTrial==NULL) {
            Error(OUT_OF_MEMORY);
            exit(OUT_OF_MEMORY);
        }
    } else {
        Error(INVALID_REQUEST_ALLOCATE_MEMORY);
        exit(INVALID_REQUEST_ALLOCATE_MEMORY);
    }
    InitialiseSetsTrial();
}

/*-----
   InitialiseSetsTrial:
   Reset gSetsTrial to initial state.
   @assume gSetsTrial
   @assume gM - the size of the trial array
   -----*/
void InitialiseSetsTrial() {
    int i;
    if (gSetTypes[0]==0) gInitTrial = 1;
    else gInitTrial = 0;
    for (i=0; i<gM; i++) {
        gSetsTrial[i] = gInitTrial;
    }
}

/*-----
   NextSetsTrial:
   Iterate SetsTrial to next valid trial.
   @assume gSetsTrial
   @assume gM - the size of the trial array
   @return success of iteration - false when no more
   -----*/
int NextSetsTrial(void) {
    int i;
    int valid;

    valid = 2;
    while (valid == 2) {
        i=gM-1;
        gSetsTrial[i]++;
        while (gSetsTrial[i]>gMaxNumOfSetTypes) {
            gSetsTrial[i]=gInitTrial;
            i--;
            if (i<0) break;
            gSetsTrial[i]++;
        }
        if (i<0) {
            valid = 0;
            break;
        }
        for (i=0; i<gMaxNumOfSetTypes; i++) gSetTypesCount[i]=0;
        for (i=0; i<gM; i++) {
            gSetTypesCount[gSetsTrial[i]]++;
        }
        for (i=0; i<gMaxNumOfSetTypes; i++) {
            if (gSetTypesCount[i]!=gSetTypes[i]) break;
        }
        if (i == gMaxNumOfSetTypes) valid=1;
    }
    return valid;
}

```



```

}
625
/*-----
    InitialiseBarMix:
    Reset sign of item to initial state.
    @param item - the item to reset
    @assume gSets
    @assume gM - the size of the trial array
    -----*/
void InitialiseBarMix(int item) {
    set_item *set;
    int i;

    set = gSets[item];
    for (i=0; i<gK; i++) {
        set[i].sign = NONBAR;
    }
    if (gSetsTrial[item] > 0) NextBarMix(item, gSetsTrial[item]);
}
640
/*-----
    NextBarMix:
    Iterate set to next valid mix.
    @param item
    @param bars - number of bars
    @assume gSets
    @assume gM - the size of the trial array
    @return success of iteration - false when no more
    -----*/
int NextBarMix(int item, int bars) {
    int i;
    int valid;
    int count;
    set_item *set;

    set = gSets[item];
    valid = 2;
    while (valid == 2) {
        i=gK-1;
        set[i].sign++;
        while (set[i].sign> 1 || set[i].value == 0) {
            set[i].sign=0;
            i--;
            if (i<0) break;
            set[i].sign++;
        }
        if (i<0) {
            valid = 0;
            break;
        }

        count = 0;
        for (i=0; i<gK; i++) {
            if ((set[i].sign == BAR) && (set[i].value != 0)) count++;
        }

        if (count == bars) valid = 1;
    }
    return valid;
}
660
670
680
/*-----
    CheckBRD:
    Check and report whether gSets are BRDs.
    Sets v, b, r, k, l are BRD if all the non-zero bar and non-bar
    differences occur the same number of times.
    BRDs attach a sign. Likes give a nonbar. Unlikes give bar.
    @return int which is true if BRD otherwise false
    @assume gSets
    @assume gResults - has been initialised
    @assume gK - the size of the results array and modulus
    @assume gM - the number of sets in gSets
    -----*/
int CheckBRD(void) {
    int i,j,m,result;
    set_item *set;
    690
    700

```

```

int occur;

/* Calculate results */
for (i=0;i<gK;i++) {
    for (j=0;j<gK;j++) {
        for (m=0;m<gM;m++) {
            set = gSets[m];
            result = set[j].value - set[i].value;
            if (result < 0) result += gV;
            if (set[j].sign == set[i].sign) {
                gResults[result].count++;
            } else {
                gResults[result].bar_count++;
            }
        }
    }
}

/* Check results - consider only non-zero results */
occur = 0;
for (i=1;i<gV;i++) {
    if (gResults[i].count>0 || gResults[i].bar_count>0) {
        if (occur == 0) occur = gResults[i].count;
        else if (gResults[i].count != occur) break;

        if (gResults[i].bar_count != occur) break;
    }
}
if (i==gV) {
    return 1;
} else {
    return 0;
}
}

/*-----
FindBRD:
Tries possible signed combinations of the SDS and tests whether
satisfy the BRD test. Eliminates the combinations that do not satisfy
the constraints.
@return int which is true if BRD found otherwise false
@assume gSets
@assume gResults
@assume gK - the size of the results array and modulus
@assume gM - the number of sets in gSets
-----*/
int FindBRD(void) {
    int i;
    int valid;

    CreateSetsTrial();
    while (NextSetsTrial()) {
        for (i=0; i<gM; i++) {
            InitialiseBarMix(i);
        }
        valid = 2;
        while (valid == 2) {
            InitialiseResults();
            if (CheckBRD()) {
                valid = 1;
                break;
            }

            i=gM-1;
            while (NextBarMix(i, gSetsTrial[i])==0) {
                InitialiseBarMix(i);
                i--;
                if (i<0) break;
            }
            if (i<0) {
                valid = 0;
                break;
            }
        }
    }
    if (valid == 1) fprintf(stdout, "Found BRD!\n");
    else fprintf(stdout, "BRD not found.\n");
}

```

Jun 18 19:19 1997

BRD.C 11

```
    return valid;  
}
```