



UNIVERSITY  
OF WOLLONGONG  
AUSTRALIA

University of Wollongong  
**Research Online**

---

Department of Computing Science Working Paper  
Series

Faculty of Engineering and Information Sciences

---

1981

# A representation approach to the tower of Hanoi problem

M. C. Er

*University of Wollongong*

---

## Recommended Citation

Er, M. C., A representation approach to the tower of Hanoi problem, Department of Computing Science, University of Wollongong, Working Paper 81-8, 1981, 13p.  
<http://ro.uow.edu.au/compsciwp/18>

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library:  
[research-pubs@uow.edu.au](mailto:research-pubs@uow.edu.au)

A REPRESENTATION APPROACH TO THE  
TOWER OF HANOI PROBLEM

M. C. ER

Department of Computing Science,  
The University of Wollongong,  
Post Office Box 1144,  
Wollongong, N.S.W. 2500  
Australia

ABSTRACT

By making the moving direction of each disc explicit in the representation, a bit-string so constructed can be used to drive the Tower of Hanoi algorithm. The behaviour of disc moves is further analyzed based on the bit-string representation. It has been shown that the bit-string for moving  $n$  discs can be used to generate successively the Gray codes of  $n$  bits.

Keywords and phases: Tower of Hanoi, representation approach, Gray codes, combinatorial algorithms.

CR Categories: 5.30, 5.39

A REPRESENTATION APPROACH TO THE  
TOWER OF HANOI PROBLEM

M. C. ER

Department of Computing Science,  
The University of Wollongong,  
Post Office Box 1144,  
Wollongong, N.S.W. 2500  
Australia.

1. INTRODUCTION

Algorithms for the Tower of Hanoi problem are often used in the introductory texts on computer programming for demonstrating the power of recursion (Hayes, 1977; Dijkstra, 1971; Dromey, 1981). Interesting though these recursive algorithms are, beginners are not always convinced that these algorithms will work until they are run. Given such recursive algorithms, it is not obvious how to move discs around until one actually steps through the programs.

Hayes (1977), Buneman and Levy (1980) present two iterative algorithms for solving the Tower of Hanoi problem, hoping that they are less mysterious than the recursive solutions. It is however not clear why the smallest disc is always moved in the cyclic order. At any rate, they did not argue that their solutions are the optimal ones; namely, to move the tower of discs from one peg to the other peg in the minimum number of steps. An analysis is clearly necessary.

It is a common experience in the Artificial Intelligence research (Korf, 1980) that a suitable representation may lead to an efficient and transparent algorithm. It is the theme of this paper to examine the Tower of Hanoi problem in this light. By encoding the disc moves into a bit-string, we show that a straight forward interactive algorithm can be constructed. More important, the bit-string lends a hand to an analysis of the behaviour of the algorithm.

## 2. THE PROBLEM

The Tower of Hanoi problem involves three pegs ( $P_1$ ,  $P_2$  and  $P_3$ ) and  $n$  discs ( $D_1, D_2, D_3 \dots D_n$ ) such that  $D_1 < D_2 < D_3 < \dots < D_n$ , where  $D_1$  is the smallest disc. Initially, all the  $n$  discs are placed on a peg  $P_i$  as a pyramid with  $D_1$  on the top. The task is to move these  $n$  discs from  $P_i$  to  $P_j$  such that  $i \neq j$ , subject to the following constraints:

- C1 : Only the top disc of a tower may be moved from one peg to the other;
- C2 : No disc may rest upon a smaller disc at any time;
- C3 : Only one disc may be moved at a time.

## 3. OTHER ITERATIVE SOLUTIONS

Hayes (1977) gives reasons to believe that the smallest-disc moves must alternate with other disc moves. However, he did not explain why the smallest disc must move cyclically. Nor did Buneman and Levy (1980) give reasons to support their algorithm that the smallest disc must always move in the clockwise direction. Clearly, to move  $n$  discs from peg 1 ( $P_1$ ) to peg 3 ( $P_3$ ), one needs not have to move  $n$  discs from  $P_1$  to peg 2 ( $P_2$ ), then move them to  $P_3$ . Buneman and Levy's algorithm therefore does not provide an optimal solution under certain circumstances.

Consider the general case. Suppose the smallest disc ( $D_1$ ) is on top of  $P_1$ , and the other two discs  $D_i$  and  $D_j$  are on top of  $P_2$  and  $P_3$  respectively. Clearly  $D_1 < D_i$  and  $D_1 < D_j$ . Suppose we have just moved  $D_1$ , either  $D_i$  or  $D_j$  will be moved in the next move. As  $D_i \neq D_j$ , the next move is a unique solution depending on which of  $D_i$  and  $D_j$  is smaller. Suppose  $D_i < D_j$ , the next move is to move  $D_i$  on top of  $D_j$ . After that, we have to move  $D_1$ . Now we have two choices; either to move  $D_1$  on top of  $D_i$  or to move  $D_1$  to  $P_2$ . It is under these circumstances that both Hayes (1977), and Buneman and Levy (1980) fail to satisfy us that one choice is better than the other. A deeper analysis is clearly called for.

#### 4. TREE AND BIT-STRING REPRESENTATION

Suppose the three pegs are arranged in a circle as shown in figure 1.

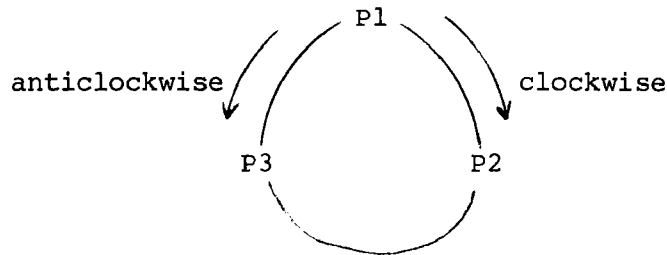


Figure 1 Arrangement of Pegs

Define the clockwise direction as  $P1 \rightarrow P2 \rightarrow P3 \rightarrow P1$ , and the anticlockwise direction as  $P1 \rightarrow P3 \rightarrow P2 \rightarrow P1$ . Any top disc can be moved to its neighbouring peg in either clockwise or anticlockwise direction at any movement. We can view the clockwise and anticlockwise moves as traversal of left and right branches of a binary tree respectively. It is then possible to represent the directions of disc moves as a binary tree. In other words, we make explicit the directions of disc moves in the representation.

Suppose we are asked to move  $n$  discs from a peg to its neighbouring peg in the clockwise direction, we can assume that the source is P1, and the destination is P2 without loss of generality. This can be expressed formally as follows:

$$D_1, D_2, \dots, D_n : P1 \downarrow P2.$$

When  $n = 1$ , only disc  $D_1$  can be moved from P1 to P2 in the clockwise direction in one step:

$$D_1 : P1 \downarrow P2.$$

When  $N = 2$ , the smallest disc  $D_1$  should be moved in the anticlockwise direction from P1 to P3. Then the larger disc  $D_2$  will be moved from P1 to P2 in the clockwise direction. After that,  $D_1$  will be moved from P3 to P1 in the anticlockwise direction thus completing the task. Namely,

$$D_1, D_2 : P1 \downarrow P2 \quad \left\{ \begin{array}{l} D_1 : P1 \downarrow P3 \\ D_2 : P1 \downarrow P2 \\ D_1 : P3 \downarrow P2 \end{array} \right.$$

The tree representations of disc moving directions are shown in figure 2.

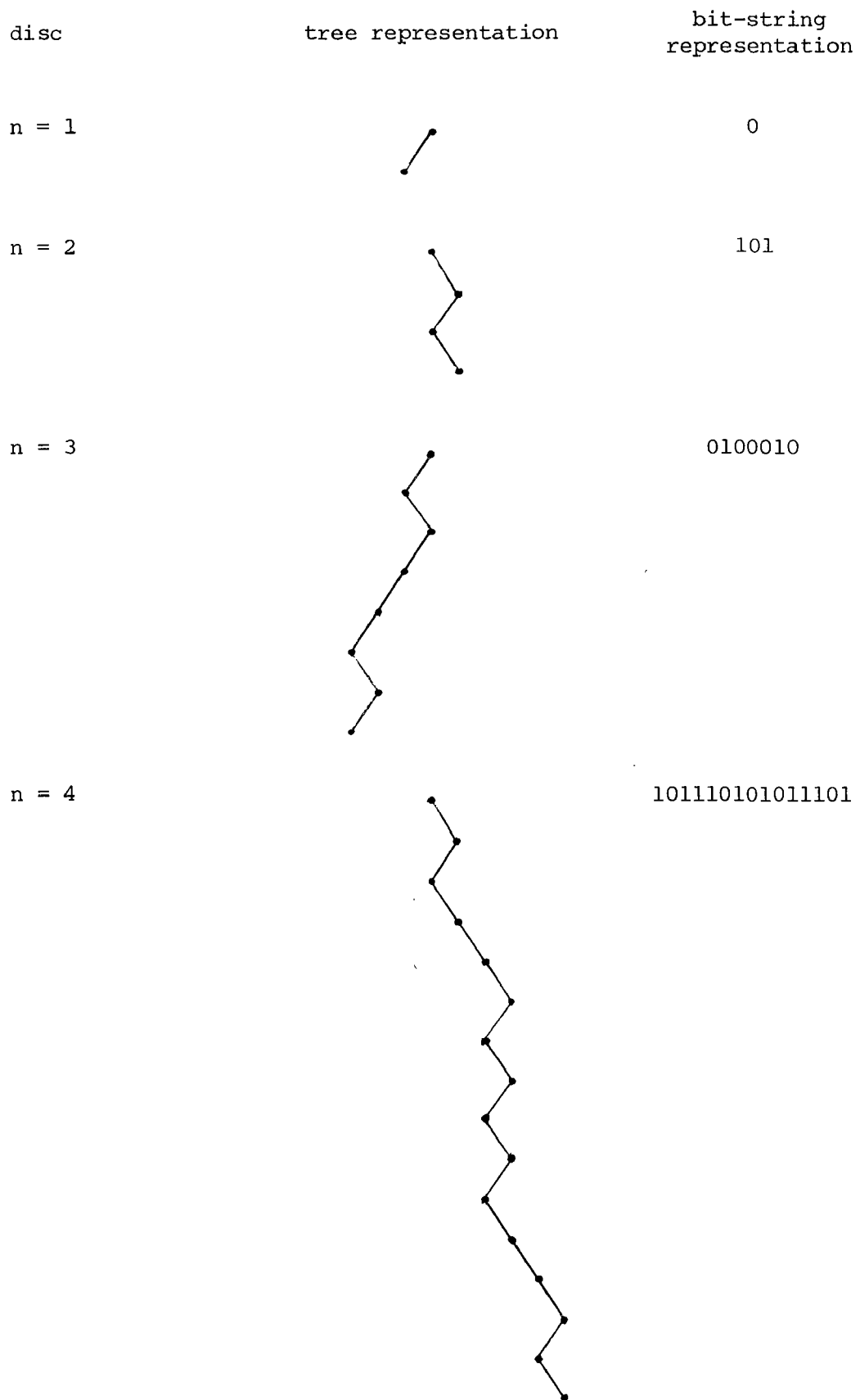


Figure 2 Moving towers clockwise

When  $n = 3$ , the steps are as follows:

$$D_1, D_2, D_3 : P1 \downarrow P2 \left\{ \begin{array}{l} D_1, D_2 : P1 \downarrow P3 \\ D_3 : P1 \downarrow P2 \\ D_1, D_2 : P3 \downarrow P2 \end{array} \right.$$

To move  $D_1$  and  $D_2$  in the anticlockwise direction, the steps can be detailed as follows:

$$D_1, D_2 : P1 \downarrow P3 \left\{ \begin{array}{l} D_1 : P1 \downarrow P2 \\ D_2 : P1 \downarrow P3 \\ D_1 : P2 \downarrow P3 \end{array} \right.$$

From the tree representation, it is apparent that the tree for moving two discs anticlockwise is the mirror image of the tree for moving them clockwise as shown in figure 3.

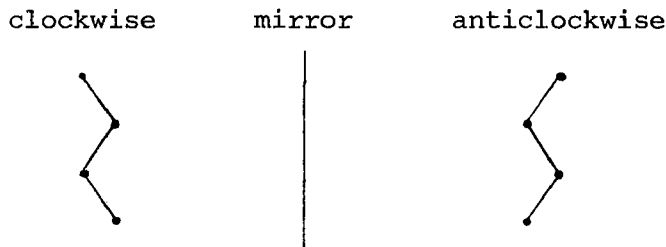


Figure 3 clockwise and anticlockwise trees

Indeed, the clockwise and anticlockwise trees are mirror images of each other is generally true for  $n \geq 1$ .

Hence, it appears that to construct the tree for  $D_1, D_2, \dots, D_{n-1}, D_n : P1 \downarrow P2$ , we simply prepend and append the mirror image of the tree for  $D_1, D_2, \dots, D_{n-1} : P1 \downarrow P2$  to the root and leaf of the following tree:



If we now encode the binary trees by using Huffman's (1952) method (namely left and right branches are represented by 0 and 1 respectively), the binary trees are collapsed into bit-strings as shown in figure 2. By virtue of the way the binary trees are constructed, we can generate the bit-strings without referring to the trees. Let  $BS(n \downarrow)$  denotes the bit-string for moving  $n$  discs clockwise.

It is obvious that

$$BS(1 \downarrow) = 1 \quad (1)$$

Let  $C(bs)$  be the one's complement of the bit-string  $bs$ . One can easily show that the mirror image of a binary tree is precisely the one's complement of its bit-string. Therefore,

$$BS(n \downarrow) = C(BS(n-1 \downarrow)) \cup C(BS(n-1 \uparrow)) \quad (2)$$

For moving  $n$  discs anticlockwise,  $D_1, D_2, \dots, D_n : P1 \downarrow P3$ , one can derive that the tree is precisely the mirror image of the clockwise case. The results are shown in figure 4.

Let  $BS(n \downarrow)$  be the bit-string for moving  $n$  discs anticlockwise. By the nature of binary digits, one can easily verify that

$$\begin{aligned} BS(n \downarrow) &= C(BS(n \uparrow)) \\ \text{or } BS(n \uparrow) &= C(BS(n \downarrow)) \end{aligned} \quad (3)$$

Finally, we establish a property of the bit-strings so generated.

Property 0 : The bit-string for moving  $n$  discs is symmetric with respect to the centre bit.

Proof : This property readily follows from (1) and (2) by induction.

[QED]



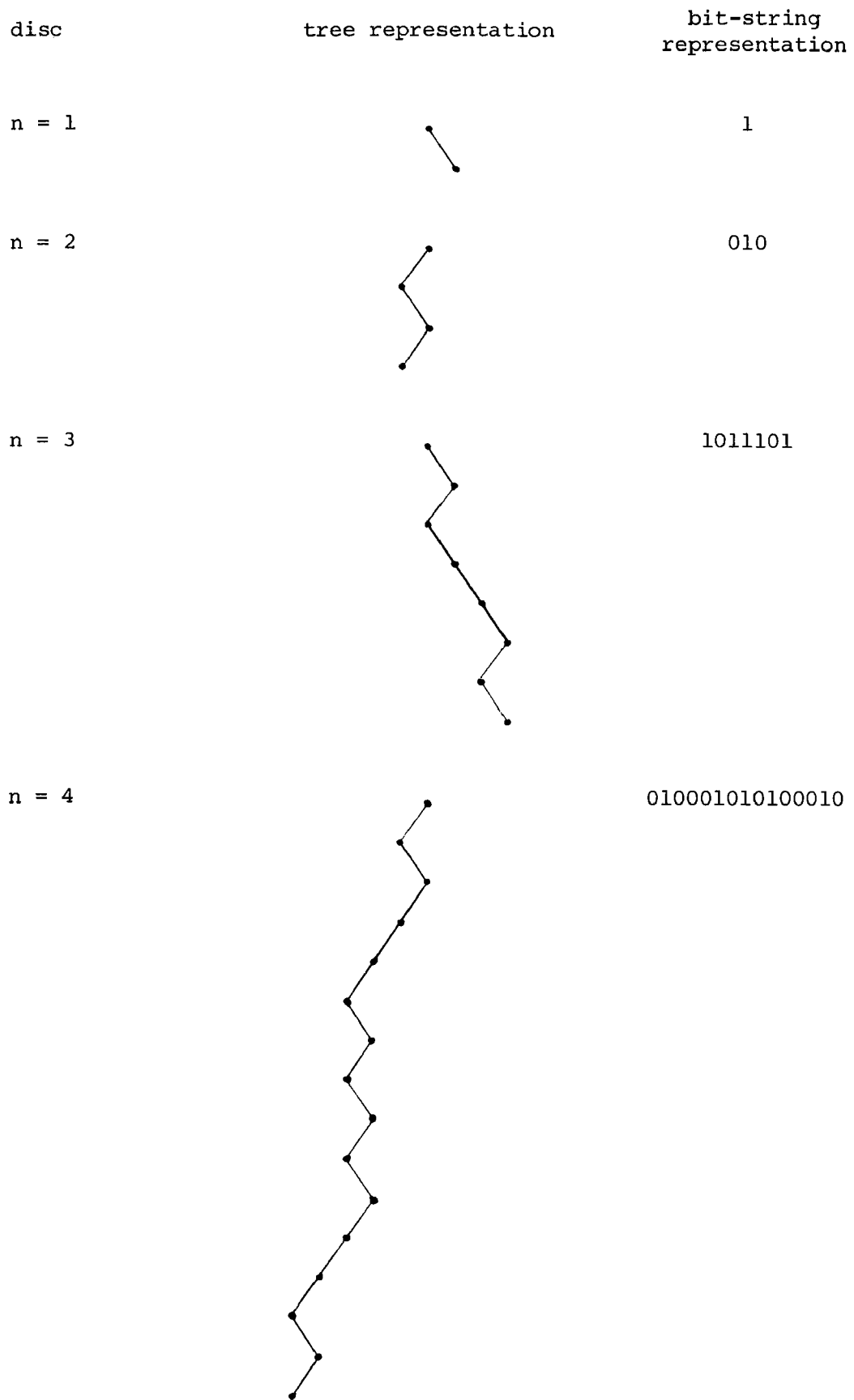


Figure 4 Moving towers anticlockwise

### 5. HOW TO MAKE MOVES

Given a bit-string for  $n$  discs, we need to be able to interpret it in order to guide the disc moves. First of all, let us establish some properties.

Property 1 :  $D_1$  is the first and last disc to move.

Proof : When  $n = 1$ , this is trivially true.

$$\begin{aligned} \text{When } n = 2, \text{ BS}(2 \downarrow) &= C(\text{BS}(1 \downarrow)) \cup C(\text{BS}(1 \downarrow)) \\ &= \text{BS}(1 \downarrow) \cup \text{BS}(1 \downarrow) \end{aligned}$$

where  $\text{BS}(1 \downarrow)$  is  $D_1$ 's move.

Likewise for  $\text{BS}(2 \downarrow)$ .

Suppose  $\text{BS}(n \downarrow)$  and  $\text{BS}(n \downarrow)$  preserve this property.

Then, by (2)

$$\text{BS}(n+1 \downarrow) = C(\text{BS}(n \downarrow)) \cup C(\text{BS}(n \downarrow))$$

We have proved by induction.

Likewise for  $\text{BS}(n+1 \downarrow)$ .

[QED]

Property 2 :  $D_1$ 's moves always alternate with the moves of other discs.

Proof : When  $n = 1$ , it is trivial.

$$\text{When } n = 2, \text{ BS}(2 \downarrow) = \text{BS}(1 \downarrow) \cup \text{BS}(1 \downarrow)$$

where  $\text{BS}(1 \downarrow)$  is  $D_1$ 's move.

This property is obviously true.

Similarly for  $\text{BS}(2 \downarrow)$ .

Suppose this property holds for  $n$  discs.

Combine with property 1, the  $D_1$ 's moves must occur at the odd positions.

Now we can prove that this property also holds for  $(n+1)$  discs.

By (2)

$$\text{BS}(n+1 \downarrow) = C(\text{BS}(n \downarrow)) \cup C(\text{BS}(n \downarrow)).$$

Note that one's complement does not change the position of its bits indicating the  $D_1$ 's moves. By virtue of the fact that  $\text{BS}(n \downarrow)$  has  $D_1$ 's moves as first and last moves, as well as at odd positions,  $C(\text{BS}(n \downarrow)) \cup C(\text{BS}(n \downarrow))$  therefore preserves property 2. Likewise for  $\text{BS}(n+1 \downarrow)$ .

[QED]

Now, we are in a position to interpret the bit-string. Let  $\{b_1, b_2, \dots, b_m\}$  be a bit-string. For a bit  $b_i$ ,  $i$  is odd,  $D_1$  is moved according to this parity: 0 and 1 are clockwise and anticlockwise respectively. For  $b_i$ ,  $i$  is even, other disc other than  $D_1$  is moved. Suppose  $D_1$  rests on  $P_1$  and  $b_i = 0$ , we cannot move the top disc from  $P_3$  to  $P_1$  due to constraint C2. Therefore, we have a unique solution, namely to move the top disc from  $P_2$  to  $P_3$ . Similar argument applies to  $b_i = 1$ . Suppose  $\text{MoveDisc}(P_i, P_j)$  is to move the top disc from  $P_i$  to  $P_j$ . The algorithm for moving  $D_i$ , such that  $i \neq 1$  is summarized in figure 5.

```
Switch on  $D_1$  into
{
  Case P1 : If  $b_i = 0$ 
             then MoveDisc (P2, P3)
             else MoveDisc (P3, P2)
             endcase

  Case P2 : If  $b_i = 0$ 
             then MoveDisc (P3, P1)
             else MoveDisc (P1, P3)
             endcase

  Case P3 : If  $b_i = 0$ 
             then MoveDisc (P1, P2)
             else MoveDisc (P2, P1)
             endcase
}
```

Figure 5 Moving other disc

## 6. FURTHER ANALYSIS

We now further analyze the bit-string for moving  $n$  discs to reveal the inherent properties of the Tower of Hanoi problem.

Property 3 : The smallest disc always moves in a cyclic order.

Proof : When  $n = 1$ , this is trivial.

Suppose this property holds for  $(n-1)$  discs.

Namely, all the odd position bits are having the same parity. By properties 1 and 2 and (2), the odd position bits of the bit-string for moving  $n$  discs again have the same parity.

[QED]

Property 4 : The solution offered by the bit-string for moving  $n$  discs is optimal.

Proof : When  $n = 1$ ,  $BS(1 \downarrow)$  for  $D_1 : P1 \downarrow P2$  or  
 $BS(1 \downarrow)$  for  $D_1 : P1 \downarrow P2$   
 is obviously optimal.

Suppose  $BS(n-1 \downarrow)$  and  $BS(n-1 \downarrow)$  are optimal.

We now show that  $BS(n \downarrow)$  is optimal too.

To move  $n$  discs from  $P1$  to  $P2$ , we need to move the top  $(n-1)$  discs from  $P1$  to  $P3$ , then move  $D_n$  from  $P1$  to  $P2$ , and finally move the  $(n-1)$  discs again from  $P3$  to  $P2$ . That is,

$$D_1, D_2, \dots, D_n : P1 \downarrow P2 \quad \left\{ \begin{array}{l} D_1, D_2, \dots, D_{n-1} : P1 \downarrow P3 \\ D_n : P1 \downarrow P2 \\ D_1, D_2, \dots, D_{n-1} : P3 \downarrow P2 \end{array} \right.$$

As  $BS(n-1 \downarrow)$  is optimal for  $D_1, D_2, \dots, D_{n-1} : P_i \downarrow P_j$ , where  $i \neq j$ . It follows that  $BS(n \downarrow)$  for the composite solution of moving  $n$  discs,  $D_1, D_2, \dots, D_n : P1 \downarrow P2$ , is optimal. Similar argument holds for  $BS(n \downarrow)$ .

[QED]

Property 5 : The optimal solution takes  $2^n - 1$  steps.

Proof : Let  $S_n$  be the number of bits in  $BS(n \downarrow)$ .

From (2),  $S_n = 2 * S_{n-1} + 1$ .

As  $S_1 = 1$  by (1). Therefore

$$\begin{aligned} S_n &= \sum_{i=0}^{n-1} 2^i \\ &= 2^n - 1 \end{aligned}$$

As a bit corresponds to a step, thus the optimal solution takes  $2^n - 1$  steps.

[QED]

Property 6 : All  $D_i$ ,  $i = \text{odd}$ , move in the same direction. Whereas, all  $D_j$ ,  $j = \text{even}$ , move in the opposite direction.

Proof : From (2),  $BS(n \downarrow) = C(BS(n-1 \downarrow)) \ 1 \ C(BS(n-1 \downarrow))$

where  $C(BS(n-1 \downarrow)) = C(C(BS(n-2 \downarrow)) \ 1 \ C(BS(n-2 \downarrow)))$

$= C(C(BS(n-2 \downarrow))) \ 0 \ C(C(BS(n-2 \downarrow)))$

As the centre bit of  $BS(n \downarrow)$  is an indication of the moving direction of  $D_n$ , so the centre bit of  $C(BS(n-1 \downarrow))$  indicates the moving direction of  $D_{n-1}$ .

As the centre bits of  $BS(n \downarrow)$  and  $C(BS(n-1 \downarrow))$  are of different parity, we have proved that  $D_i$  and  $D_j$ , such that  $|i-j| = 1$ , move in opposite directions.

Furthermore,

$$\begin{aligned} C(C(BS(n-2 \downarrow))) &= BS(n-2 \downarrow) \\ &= C(BS(n-3 \downarrow)) \ 1 \ C(BS(n-3 \downarrow)). \end{aligned}$$

As the centre bits of  $BS(n \downarrow)$  and  $C(C(BS(n-2 \downarrow)))$  are having the same parity, we thus prove that  $D_n$  and  $D_{n-2}$  are moving in the same direction.

By induction, the property holds.

[QED]

Property 7 : Let  $M_i$  be the number of steps taken by  $D_i$  to get to the destination in the optimal solution. Then  $M_i = 2^{n-i}$ .

Proof: As we know, the centre bit of  $BS(n \downarrow)$  indicates the moving direction of  $D_n$ , and that is the only bit to do so. Hence,  $M_n = 1$ .

Suppose,  $M_i = 2^{n-i}$  for  $0 < i \leq n$ .

By (2),  $BS(i \downarrow) = C(BS(i-1 \downarrow)) \mid C(BS(i-1 \downarrow))$ .

$$\begin{aligned} \text{Thus, } M_{i-1} &= 2 * 2^{n-i} \\ &= 2^{n-(i-1)} \end{aligned}$$

[QED]

## 7. IMPLEMENTATIONS

Now we are in a position to implement the algorithm. From property 5, we know that it takes  $2^n - 1$  steps to move  $n$  discs from a peg to a target peg. So the control loop can be implemented as a for-loop. Further, we know from property 2 that the moves of  $D_1$  alternate with other disc moves. Therefore, the body of the for-loop comprises two move-disc instructions; one for moving  $D_1$ , another for moving other discs. A program based on these ideas has been written using C, and is included in Appendix A for reference. Notice that the generating function, Generate, successively generates the bit-string from 1 disc up to  $n$  discs based on (2).

A moment's reflection would convince us that to generate the bit-string for  $n$  discs, it is not necessary to generate all the bit-strings for 1 disc up to  $n$  discs. We can indeed generate the bit-string for  $n$  discs straight away, by taking the advantage of property 6. Before we spell out the details of the direction generating function, we prove a property first.

Property 8 : All bit positions  $2^{i-1} + 2^i * j \leq 2^n - 1$

where  $j=0,1,2,\dots$  and  $i=\text{even}$ , are occupied by bits of same parity. Whereas, other bit positions,

$p \neq 2^{i-1} + 2^i * j$ , are occupied by bits of opposite parity.