中　華　大　學

碩 士 論 文

平行移動「河內塔問題」之反覆式演算法分析

Analysis on an Iterative Algorithm of "The Tower of Hanoi problem" with Parallel Moves

系 所 別 ： 資訊工程學系碩士班

學號姓名 ： E08902018 王 裕 國

指導教授 ： 吳 哲 賢 博 士

中 華 民 國 九 十 二 年 一 月

平行移動「河內塔問題」之反覆式演算法分析

研究生：王裕國　　　指導教授：吳哲賢 博士

中華大學資訊工程研究所

中 文 摘 要

「河內塔問題」是一個古老而且著名的數學問題，1883 年由 Lucas 提出以後，距今已超過 100 年，一直受到廣泛討論與研究。1971 年，Dijkstra 首先提出「河內塔問題」之遞迴式演算法的最佳解。接著 Atkinson 在 1981 年提出一個新的變形問題：「循環式河內塔問題」，並提出遞迴式演算法的最佳解。1992 年，Wu 和 Chen 提出另外一種新的變型問題：「平行式河內塔問題」，與其遞迴式最佳解；隨後於 1993 年，又提出了一個複合式變形：「平行循環式河內塔問題」及其遞迴式演算法的最佳解。

本論文主要討論及分析「河內塔問題」其反覆式演算法之最佳解。在 1980 年，Buneman 提出了「河內塔問題」之反覆式演算法最佳解，接著於 1983 年 Walsh 對「循環式河內塔問題」也提出反覆式演算法的最佳解。我們在本論文中將針對 「平行式河內塔問題」，設計出一種簡單且有效率的反覆式演算法，並證明其搬動次數為最佳解。

Analysis on an Iterative algorithm of "The Tower of Hanoi problem" with Parallel Moves

Student：Yu-Kuo Wang Advisor：Dr. Jer-Shyan Wu

Institute of Computer Science and Information Engineering
Chung Hua University

English Abstract

"The Tower of Hanoi problem" is an ancient and famous mathematical problem. It has over 100 years since Lucas presented in 1883. In 1971, Dijkstra first presented the optimal solution. And then, Atkinson in 1981 proposed a variant, known as "The cyclic Tower of Hanoi problem" and its recursive optimal solution. In 1992, Wu and Chen propose another new variant： "The parallel Tower of Hanoi problem" and its recursive solution; continuing in 1993, proposed another compose variant: "The cyclic parallel Tower of Hanoi problem" and its recursive solution.

In this thesis we are aimed at this problem to discuss the iterative optimal solutions. In 1980, Buneman presented the iterative optimal solutions for the problem. And then, Walsh presented the iterative optimal solutions for "The cyclic Tower of Hanoi problem" in 1983. We will design a simple and effective iterative algorithm to implement "The parallel Tower of Hanoi problem", and prove its moving is an optimal solution.

Acknowledgements

First, I would like to express my great gratitude to my advisor Dr. Jer-Shen Wu for his instruction and encouragement, whose kind encouragements and valuable suggestions have always been a source of faith and strength for me.

And then, I want to thank all my fellow classmates in the studying carrier. Especially thank my dear friend: Sheng-Hung Yi for his assistance and discussion in programming.

Finally, I dedicate this thesis to my wife, Miss Kuo-Ying Hsiung. Without her encouragement and given support to me with endless love, it is impossible to finish this paper.

Contents

List of figures

Chapter 1

Introduction

1.1 Ancient Legend

"The Tower of Hanoi problem" (TTOHP) was invented by the French mathematician Edouard Lucas in 1883 base on an ancient legend of Indian, so the legend goes, monks in a temple have to move a pile of 64 sacred golden disks from one peg to another peg. There are three pegs (origin peg, destination peg, intermediate peg), 64 disks of different sizes are placed in small-on large order disk on source peg. The problem is to finding the minimum number of moves in which the disks can be transferred to destination peg in origin order. The rule of disks movements are follows：

Rule1. Only one of the topmost disks can be moved at a time.

Rule2. No disk can be paced on a smaller one.

According to the legend, monks start moving disks back and forth, between three pegs they have to work day and night to solve the puzzle. When they finish their work, the temple will crumble into dust and the world will go to end.

Figure 1.1 Tower of Hanoi Puzzle

1.2 History

There isn't a general solution until 1941 Stewart and Frame [53] were given the solution. There existed a very elegant recursive solution for the standard TTOHP problem. An earlier version of the recursive solution is discussed by Dijkstra [6] in 1971, and a slightly different form is described by Hayes [33] in 1977. A variant, known as the cyclic Tower of Hanoi problem was propose by Atkinson [2] in 1981. Er [8-29] describes one of the most intricate yet challenging problems： The generalized color Towers of Hanoi problem, and present a simple recursive solution to it.

An algorithm for 4-peg and 5-peg tower is discussed by Er, Majumdar[37-46], Hinz[34], and Eggers [7]. This problem （Reve's Puzzle） is still open today, even that seven different approaches to the muti-peg Tower of Hanoi problem are all equivalent.

Wu and Chen [57,58] propose another two variant 3-peg forms ： Parallel and cyclic parallel moves in 1992 and 1993. They also represent recursive solutions. Various iterative solutions have been discovered by Buneman&Levy [3], Dijkstra

[6], Hayes and Walsh [55, 56], Er [8-29], Rohl [51], Gedeon [31, 32], Pettorossi [50], Allounche [1], Majumdar [40,41], Chedid & Mogi [5], Lu and Dillon [35] etc. But all is limited in standard and cyclic moves form.

The multi-peg ($k \geqq 4$) TTOHP is another generalized version of this classical problem. It is widely discussed, but is still open today. Newman-Wolfe [49] obtained some lower and upper bounds on the number of moves for different ranges of the number of pegs. They had given a recursive formulation for computing the number of moves. But interestingly, it is not known whether the number of moves given in Boardman's formulation is optimal and no evidence to contrary is available either.

In this paper we will propose our iterative algorithm to implementation "The Tower of Hanoi problem" with parallel moves.

1.3 Outline of the Paper

In this paper, we pay our attention to the iterative algorithm on parallel move. The thesis is dividing into seven chapters. In chapter1 we introduce the definition of "The Tower of Hanoi problem" (TTOHP), and the history of solving course.

Chapter2 is representation original problem of TTOHP and its general solution including Hanoi Graph. In chapter 3, section 3.1 is describe the recursive algorithm for solve TTOHP with cyclic moves. Section 3.2 is describe the recursive algorithm for solve TTOHP with parallel moves. Section 3.3 is describe the recursive algorithm for solve TTOHP with cyclic parallel moves.

Chapter4 describe the iterative implementation for TTOHP. Chapter5 describe the iterative implementation for TTOHP with cyclic moves.

In chapter6 is representation our observation: iterative algorithm for TTOHP with parallel moves. Chapter7 is the conclusion, including our research and studying topic in the future.

Chapter 2

Recursive Algorithm on "The Tower of Hanoi problem"

2.1 The Problem and Its Origin

In the first we simply TTOHP as following：

[**Definition 1**] ：$p_1, p_2, p_3, \ldots, p_k,$ denote k pegs, $d_1, d_2, d_3, \ldots, d_n,$ denote n disks

[**Definition 2**] ：An optimal solution for f(k, n) is a sequence of move n disks from source peg transfers to destination peg in origin order with k pegs

The traditional algorithm （recursive）for solving f(3,n) problem from follow How many moves will it take to transfer n disks from the source peg to the destination peg?

A. Recursive pattern

algorithm：

To move n disks from A to B using C as spare:

1、If n is 1 ， just do it.

2、If n>1

(1)、Move the top n-1 disks from A to C using B as spare

(2)、Move the bottom disks from A to B

(3)、Move n-1 disks from C to B using A as spare

We will see some real case by paragraph.

1 disk: 1 move

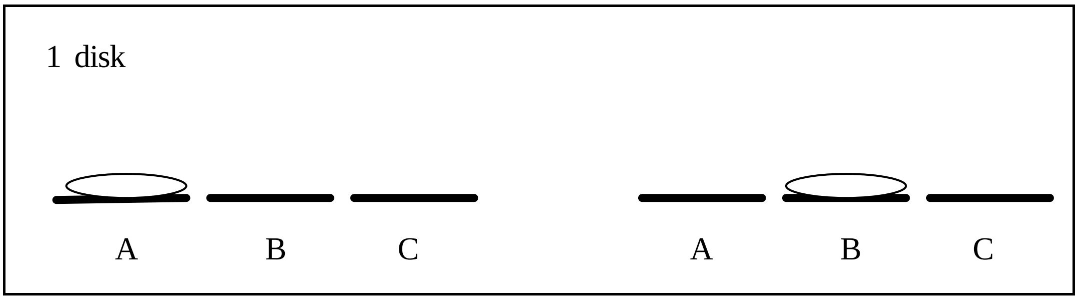


Figure 2.1a One disk moving

2 disks: 3 moves

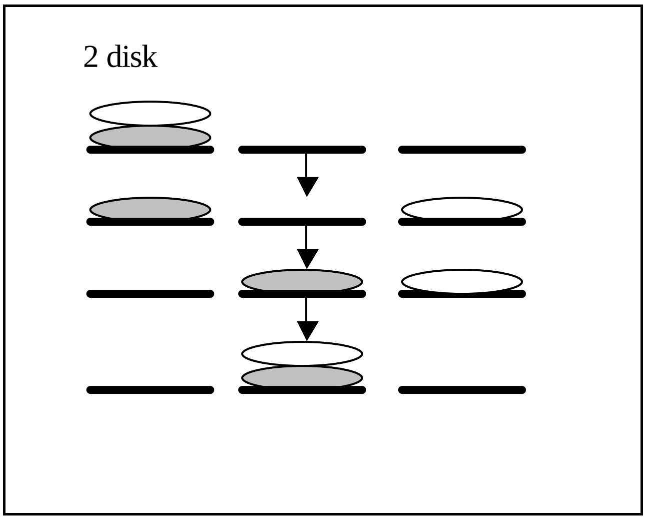


Figure 2.1b Two disks moving

3 disks: 7 moves

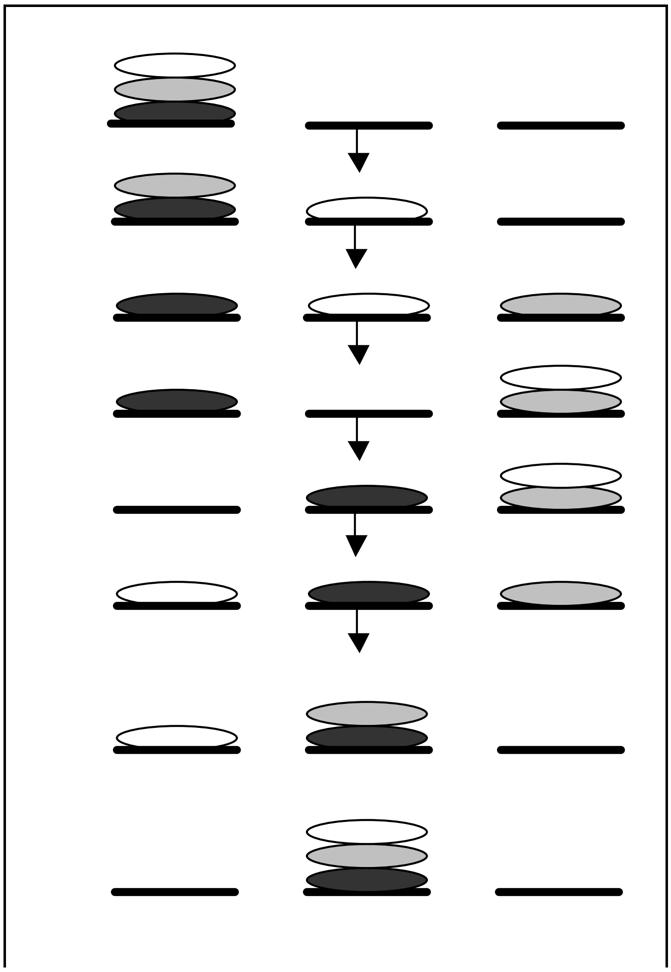


Figure 2.1c Three disks moving.

Can we work through the moves for transfer **4 disks**? It should take 15 moves. How about **5 disks, 6 disks**? Do you see a pattern?

From the moves necessary to transfer one, two, and three disks, we can find a *recursive pattern* - a pattern that uses information from one step to find the next step - for moving n disks from peg A to peg B:

1. First, transfer n-1 disks from peg A to peg C. The number of moves will be the same as those needed to transfer n-1 disks from peg A to peg C. Call this number M moves. [As you can see above, with three disks it takes 3 moves to transfer two disks (n-1) from peg A to peg C.]
2. Next, transfer disk n to peg B [1 move].
3. Finally, transfer the remaining n-1 disks from peg C to peg B. [Again, the number of moves will be the same as those needed to transfer n-1 disks from peg A to peg C, (or M moves).]

Therefore the number of moves needed to transfer n disks from peg A to peg B is **2M+1**, where M is the number of moves needed to transfer n-1 disks from peg A to peg B.

Unfortunately, if we want to know how many moves it will take to transfer 100 disks from peg A to peg B, we will first have to find the moves it takes to transfer 99 disks, 98 disks, and so on. Therefore the recursive pattern will not be much help in finding the time it would take to transfer all the disks.

However, the recursive pattern can help us generate more numbers to find an explicit (non-recursive) pattern. Here's how to find the number of moves needed to transfer larger numbers of disks from peg A to peg B, remember that M is the number of moves needed to transfer n-1 disks from peg A to peg C:

1. for **1 disk** it takes 1 move to transfer 1 disk from peg A to peg C;
2. for **2 disks**, it will take 3 moves: 2M + 1 = 2(**1**) + 1 = **3**
3. for **3 disks**, it will take 7 moves: 2M + 1 = 2(**3**) + 1 = **7**
4. for **4 disks**, it will take 15 moves: 2M + 1 = 2(**7**) + 1 = **15**
5. for **5 disks**, it will take 31 moves: 2M + 1 = 2(**15**) + 1 = **31**
6. For **6 disks**...?

B. Explicit Pattern

Number of Disks	Number of Moves
1	1
2	3
3	7
4	15
5	31

Powers of two help reveal the pattern:

Number of Disks (n)	Number of Moves
1	$2^1 - 1 = 2 - 1 = 1$
2	$2^2 - 1 = 4 - 1 = 3$
3	$2^3 - 1 = 8 - 1 = 7$
4	$2^4 - 1 = 16 - 1 = 15$
5	$2^5 - 1 = 32 - 1 = 31$
.	
.	

So the formula for finding the number of steps it takes to transfer n disks from peg A to peg B maybe **2^n – 1 times**. Next we will prove it is the optimal solution.

Theorem 1：For every k=3, n>0, there is a solution for f (n, k)

proof：1. for n=1 ， f(3,1)=1 ， it is trivial

2. From above algorithm statement we can easily prove this by induction

Corollary 1. The optimal solution for f（3,n） $=2^n-1$

Proof : 1. for n=1， f（3,1） $=1=2^1-1$ is trivial

2. for n>1， suppose f（3,k） $=2^k-1$ is hold

Then f(3, k+1)=2 f（3,k） +1

$=2*(2^k-1)+1$

$=2^{k+1}-2+1$

$=2^{k+1}-1$

From induction and above discursion we have prove the optimal solution of TTOHP for f（3,n） is 2^n-1 times moves

From this formula you can see that even if it only takes the monks one second to make each move, it will be $\mathbf{2^{64}-1}$ second before the world will end. This is about 590,000,000,000 years (that's 590 billion years) - far, far longer than some scientist' s estimate the solar system will last. That's a really long time! 。

2.2 Hanoi Graph

In 1983 Er[16] presented a state-space graph （Hanoi graph） for representing the states and their transitions of n disks on three pegs is formulated. It is then transformed to a shortest path in transferring n disks in any configurations to a specified peg. The shortest-path tree clearly characterizes the generalized TTOHP; and its use leads to a very simple analysis of the generalized problem. The best-case, the average-case, and the worse-case complexities are analyzed.

A configuration of disks on the pegs can be conceptualized as a state in the solution space. The movement of a disk from one peg to another peg is there a transition of a state to another state. If a state is represented as a node, a transition of states could be represented as a link joining two states. Since a transition is reversible in the generalized problem, the corresponding link is undirected. By representing all possible states and their transitions in this way, the result is an undirected graph, which is called the state-space graph. An example of such a state-space graph for the three－disk TTOHP may be seen in figure 2.2a. Here we denote the three pegs as A, B and C. We further assume that the disks of increasing sizes are numbered successively with the smallest disk being numbered 1. Suppose disks 1, 2, and 3 are on pegs A, C, and B respectively; we may write ACB as a shorthand for representing this state. Define an admissible configuration as a configuration of the disks on the three pegs such that none of the restrictions described in the previous section is violated. From the discussion above, any admissible configuration of n disks has a unique name. Conversely, any letter sequence of length n composed of A, B, and C describes an admissible configuration uniquely. Consequently, the number of states in a state—Space graph is equal to the number of different letter sequences, i.e:3^n.

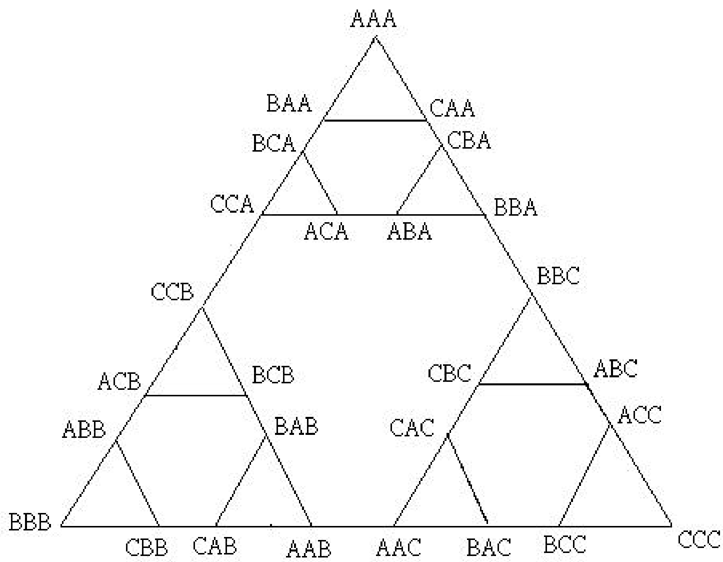


Figure 2.2a A state-space graph for the generalized TTOHP with three disks

Further, the number of links connecting to a node is either two or three; there are two links when all disks are stacked on one peg corresponding to two possible moves of the smallest disk. Three 1inks arise from a possible move of the smaller topmost disk among the two pegs not occupied by the smallest disk, in addition, to two possible moves of the smallest disk. Any other links will lead to a violation of the restrictions, and therefore should be prohibited. Thus, each of the three vertices of a state-space graph has two links, and each of other nodes has three links.

Suppose disk 1 is removed; then disk 2 becomes the smallest disk. All possible moves of disk 2 in a configuration must be identical to that of disk 1 in the similar situation. Equivalently, the topology of 1inks representing the movements of disk 2 when all smallest triangles representing the movements of disk 1 are shirked into nodes ought to be similar to the topology of a smallest triangle, because disk 2 now takes the place of disk 1. Such a topology can be recursively applied to the movements of other disks; and hence the structure of a state-space graph is recursive and simple.

Next, we discuss a symmetry property. Suppose B and C are interchanged for all state names in the state-space graph but keeping A invariant, then half of the states could be derived from another half. If a vertical 1ine is drawn passing through vertex AAA (See Figure 2.2a), the state-space graph is symmetric with respect to this vertical 1ine. The same arguments apply to the similar axes passing through the other two vertices; so the mirror symmetries embedded in the state-space graph may be easily seen. Further, Suppose A, B, and C are replaced by B, C, and A respectively for all state names, and the new state-space graph could be obtained by a 120 degree rotation of the previous state-space graph (see Figure 2.2a). Thus the state-space graph maintains the rotational symmetry as well.

The minimum number of disk move in transforming a configuration to another configuration as given by the shortest path between two nodes representing these two configurations. The shortest path between two nodes may not be unique. An example is the shortest paths between nodes BCC and CBC (see figure 2.2a)-both BCC-ACC-ABC-CBC and BCC-BAC-CAC-CBC are equally possible.

The goal of the generalized problem is to move all disks to a specified peg; such a goal node is one of the three extreme vertices of the state-space graph. If such a goal node is treated as a root and all non-shortest paths between the goal node and other nodes are removed, the result is a tree - we call it the *shortest-path tree*. It is a tree because the shortest paths between the goal node and any nodes are unique. The uniqueness follows from the mirror-symmetry property of the state-space graph - the mirror-symmetry axes passing through the vertices do not pass through any other nodes. In a further generalization, if anyone of the nodes could be the goal node, the uniqueness property of the shortest-path between a node and the goal node does not hold. But we shall not consider this extension further.

Because of the symmetry property embedded in the original state-space graph, the shortest-path tree is also symmetric. An example of the shortest-path tree, a transformation of the state-space graph shown in figure 2.2a, for representing the state-changes in moving three disks to peg *A* with the minimum numbers of steps is displayed in figure 2.2b Comparing figures 2.2a and 2.2b, we see that the non-shortest-path links that have been removed are precisely those horizontal links. This is not accidental - to move from a node of the right subtree of the shortest-path tree to its root, it is faster by ascending the right subtree directly than by ascending the left subtree, and vice-versa – a consequence of the mirror-symmetry property. Thus we have a trivial way of transforming a state-space graph to its corresponding shortest-path tree.

The most important property of the shortest-path tree is its recursive structure. Let T(n) be the shortest-path tree representing the movements of n disks. Define T(O) as a single-node tree because no disk needs to be moved in order to attain the goal state. A way of constructing T(n + 1) is to append two T(n)'s to the lowest left-most and the lowest right-most leaves of a *T(n)* respectively. This defines the recursive structure of the shortest-path tree. An instance may be seen in figure 2.2b.

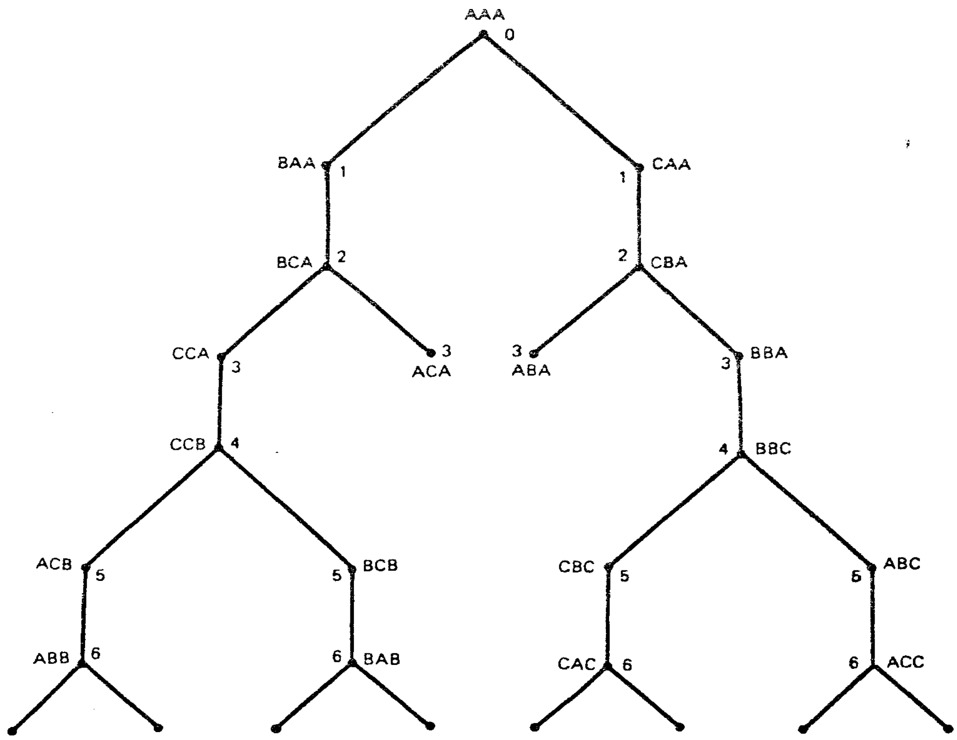


Figure 2.2b A shortest-path tree for the generalized Towers of Hanoi problem with three disks.

Chapter 3

Recursive Algorithm of Three Variants

3.1 Cyclic Moves of "The Tower of Hanoi problem"

In this section we describe the modified form of TTOHP.

We have following rules.

1. The moving direction of a disk must be clockwise.
2. Only the top disk of a tower may be moved at a time.
3. No disk can be placed on a smaller one.

[**Definition 3**] Clockwise if move direction is A→B→C→ A

[**Definition 4**] Anticlockwise if move direction is A→C→B→ A

Algorithm:

R(A(1,2,3….,n), B(0), C(0))

$\Downarrow$ a (n-1)

R(A(,n), B(0), C(1,2,3….,n-1))

$\Downarrow$ 1

R(A(0), B(n), C(1,2,3….,n-1))

$\Downarrow$ a (n-1)

R(A(0), B(1,2,3….,n), C(0))

Figure 3.1a Clockwise moving

R (A (1,2,3….,n), B(0), C(0))

$\Downarrow$ a (n-1)

R (A (n), B(0), C(1,2,3….,n-1))

$\Downarrow$ 1

R(A(0), B(n), C(1,2,3….,n-1))

$\Downarrow$ c (n-1)

R(A(1,2,3….,n -1), B(n), C(0))

$\Downarrow$ 1

R(A(1,2,3….,n-1), B(0), C(n))

$\Downarrow$ a (n-1)

R(A(0), B(0), C(1,2,3….,n))

Figure 3.1b Anti-clockwise moving

For n=3 we have

$$\begin{cases} c(n) = 2a_{n-1} + 1 \\ a(n) = 2a_{n-1} + c_{n-1} + 2 \end{cases}$$

Solve this equation we have the following solution

$$\begin{cases} c(n) = [(\sqrt{3}+3)^{n+1} - (\sqrt{3}-3)^{n+1}]/6 - 1 \\ a(n) = [(\sqrt{3}+3)^{n+2} - (\sqrt{3}-3)^{n+2}]/12 - 1 \end{cases}$$

Where c (n) is the minimum number of disk moves require to transfer n disks to the next position clockwise

a (n) is similarly but anticlockwise

Some move steps are list on Figure-7.1 for n from 1 to16.

3.2 Parallel Moves of "The Tower of Hanoi Problem"

In this second we will discuss another variant of TOH allowing parallel moves.

Rule 1: Every top disk can be simultaneously moved from origin peg to the next peg at a time.

Rule 2: No disk is ever placed upon a smaller one.

[**Definition 5**]:

A (0): peg A with no disk; similarly for B (0) and C (0).

R (A, B, C): state of pegs A, B and C

C (n): the minimal number of disk moves required to transfer n disks to the Next position clockwise (A→ B→C→A).

A (n): the minimal number of disk moves required to transfer n disks to the Next position anticlockwise (A→C→B→A).

There are four types of moves show in definition 5.

First we introduce some notations and definitions as following：

A,B,C ：A is from peg, B is to peg, C is spare peg

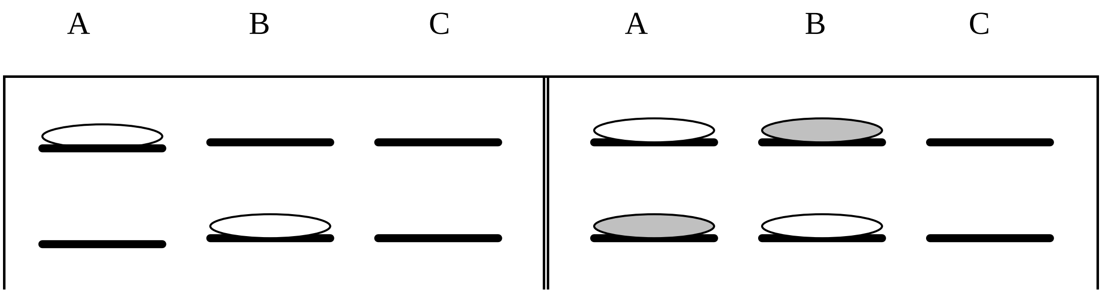
A(d1 ,d2,…)：Peg A with d1, d2 …..From top to bottom; similarly for B, C

A（0）：Peg A with no disk similarly for B, C

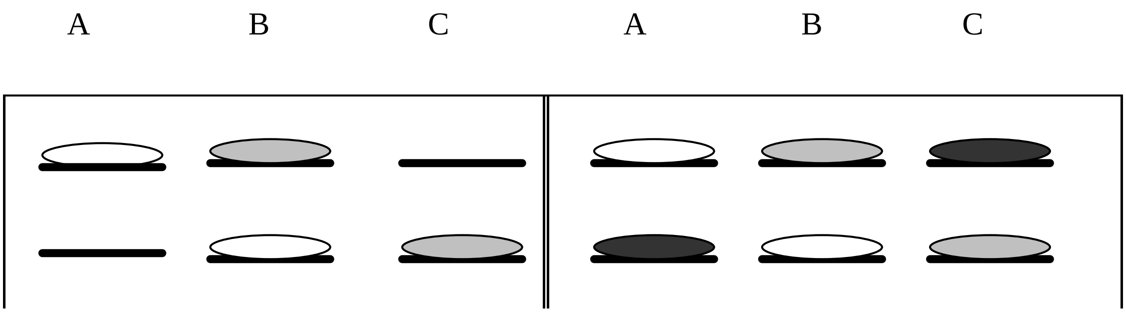
R（A,B,C）：State of pegs A, B, C

f（n）：The optimal moves for n disks

[**Definition 6**]：



(a) single move (b) exchang



(c) Consecutive move (d) Circular move

Figure 3.2a Definition of four types moving

LEMMA 3.1：Transform R(A(1,…..n), b(0),C(0)) into R(A(0), B(1,….. n), C(0)) for n = 4 is at least **2f(n-2)**+1

proof：

The transformations of R(A (1,….. n), b (0),C (0)) into R (A (0), B (1,…..,n),C (0)) can be divide into three steps :

Step 1: Transform R(A(1,...,n), B(0), C(0)) into the state R(A(n), B(0), C(1,...,n-2, n-1))

Step 2: The move of disk n from peg A to peg B

Step 3: Transform R(A(0), B(n), C(1 .... , n -1)) into R(A(0), B(1,2..., n -1,n), C(0));

Where the number of disk move for step1is at least f (n-2), (i.e. at least n-2 disks are on peg C). Step 2 takes one move and step 3 is similar to step 1.

Therefore the least number of disks moves for this problem is ***2f (n-2) +1****, for n =4*

LEMMA 3.2：To transform R(A(1,…..,n), b(0),C(0)) into R(A(0), B(1,…..,n),C(0)) for n = 4 takes exactly **2f(n-2)+1** disk moves .

Proof:

We can divide 5 steps to show this problem

Step 1: transform R(A(1,...,n), B(0), C(0)) into R(A(n- 1, n), B(1), C(2,..., n - 2));

Step 2: transform R(A(n - 1, n), B(1), C(2,..., n- 2) into R(A(n), B(n- 1), C(1,...,n-2));

Step 3: transform R(A(n), B(n - 1), C(1 .... , n -2)) into R(A(n - 1), B(n), C(1,..., n -2));

Step 4: transform R(A(n - 1), B(n), C(1,..., n -2)) into R(A(1), B(n - 1, n), C(2,..., n -2));

Step 5: transform R(A(1), B(n - 1, n), C(2,...,n - 2)) into R(A(0)), B(1,..., n), C(0))

We can form these as following:

R(A(1,2,3...,n), B(0), C(0))
⬇f(n-2)-1
R(A(n-1,n), B(1), C(2,….,n-2))
⬇ 1 (i.e. consecutive move)
R(A(n), B(n-1), C(1,2,….,n-2))
⬇ 1 (i.e. exchange move)
R(A(n-1), B(n), C(1,2,….,n-2))
⬇ 1 (i.e. consecutive move)
R(A(1), B(n-1,n), C(2,….,n-2))
⬇ f (n-2)-1
R(A(0), B(1,2,….,,n), C(0))

Figure 3.2b **T**ransform R(A(1,...,n), B(0), C(0)) into R（A(0), B(1,2,….,,n), C(0)）

***So the disk move for this algorithm is* [f(n-2)-1]+1+1+1+[f(n-2)-1]=2f(n-2)+1**

Applying the above two lemmas, we have following theorem.

THEOREM 3.3 To transform R(A（1,…..,n）, b（0）,C（0）) into R(A（0）, B（1,…..,n）,C（0）) for n = 4 the optimal algorithm is

Step 1: transform R(A(1,...,n), B(0), C(0)) into R(A(n- 1, n), B(1), C(2,..., n - 2));

Step 2: transform R(A(n - 1, n), B(1), C(2,..., n- 2) into R(A(n), B(n- 1), C(1,...,n-2));

Step 3: transform R(A(n), B(n - 1), C(1 .... , n -2)) into R(A(n - 1), B(n), C(1,..., n -2));

Step 4: transform R(A(n - 1), B(n), C(1,..., n -2)) into R(A(1), B(n - 1, n), C(2,..., n -2));

Step 5: transform R(A(1), B(n - 1, n), C(2,...,n - 2)) into R(A(0)), B(1,..., n), C(0))

And

$$f(n)=\begin{cases} 3*2^{(n-1)/2}-1 & \text{For n is odd} \\ 2*2^{n/2}-1 & \text{For n is even} \end{cases}$$

proof：

It is trivial for f(1)=1 , f(2)=3 , f(3)=5

From above two lemma f(n)=2f(n-2)+1

f(n)= $3*2^{(n-1)/2}-1$ for n is odd

f(n)= $2*2^{n/2}-1$ for n is even

3.3 Cyclic Parallel Moves of "The Tower of Hanoi Problem"

This section describe a combined variant – TTOHP problem with cyclic parallel moves, suppose there are three pegs (A,B,C) , and n disks of different size are placed in small-on-large ordering on peg source peg A. The object is to move all the n disks from peg A to either B or C in origin order by following rules ：

Rule 1: Every top disk can be simultaneously moved from origin peg to the next peg in clockwise direction A→B→C→ A, at a time

Rule 2: No disk is ever placed upon a smaller one.

There are three moves (a) single move, (b) consecutive move, (c) circular move, are similar to definition 5

Lemma 3.4. For n = 1, the minimal number of disk moves for transforming R(A(2,..., n), B(1), C(0)) into R(A(0), B(1,..., n), C(0)) is c(n)-1.

Proof:

For n = 1, consider the optimal transformation of R(A(1,...,n), B(0), C(0)) into

R(A(0), B(1 .... , n), C(0)). The first disk move is R(A(1, ... n), B(0), C(0)) →R(A(2,...,n), B(1), C(0)). Hence the minimal number of disk moves for transforming R(A(2,..., n), B(1), C(0)) into R(A(0), B(1,..., n), C(0)), is **c(n)-1.**

Lemma 3.5 For n= 1, the minimal numbers of disk moves are c(n) - 1, for the following transformations.

(1)transforming R(A(1,..., n), B(0), C(0)) into R(A(1) , B(2,..., n), C(0)),

(2)transforming R(A(2,..., n), B(1), C(0)) into R(A(0). B(1,..., n), C(0));

And a(n)- 1, for the following transformations:

(1)transforming R(A(1,..., n), B(0), C(0)) into r(A(0). B(1), C(2,...,n))

(2) transforming R(A(2, . . . , n), B(1), C(0)) into R(A(0), B(0), C(1,...,n)).

Proof:

By Lemma 3.4, this lemma is obviously shown.

Lemma 3.6 For n= 3, consider the optimal transformation of (A(1,….,n) ,B(0) ,C(0)) into R(A(0), B(1,..., n), C(0)); the details of disk moves are shown in Figure.3.3c.

Proof：

For n = 3, the transformation of R(A(1,.,., n), B(0), C(0)) into R(A(0),B(1,..., n), C(0))can be divided into three steps:

Step 1 shows the transformation of R(A(1,...,n, B(0), C(0)) into the state prior to disk n being moved from peg A to peg B .

Step 2 shows the move of disk n from peg A

Step 3 shows the transformation when disk n has been moved from peg B into R(A(0), B(1,..., n), C(0)).

In the final state of Step 1, only disk n is on peg A, and disk 1 or disk 2 or none is on peg B while the others are on peg C. And in the initial state of Step 3, only disk n is on peg B, and disk 1 or disk 2 or none is on peg A while the others are on peg C. Hence the transformation diagram is shown in Figure-3.3c

Lemma 3.7. For n = 3, consider the optimal transformation of R(A(1,...,n), B(0), C(0)) into R(A(0), B(0), C(1,...,n)); the details of disk moves are shown in Figure-3.3d.

Proof:

The transformation R(A(1,...,n), B(0),C(0)) into R(A(0), B(1,...,n), C(0)) can be divided into five steps:

Step 1 shows the transformation of R(A(1,...,n), B(0), C(0)) into the state prior to disk n being moved from peg A to peg B .

Step 2 shows the move of disk n from peg A to peg B.

Step 3 shows the transformation of the state when disk n has been moved from peg A to peg B into the state prior to disk n being moved from Peg B to peg C,

Step 4 shows the move of disk n from peg B TO peg C.

Step 5 shows the transformation of when disk n has been moved from peg C into R(A(0), B(0), C(1,..., n)).

First, in the final state of Step 1, only disk n is on peg A, and disk 1 or disk 2 or none is on peg B while others are on peg C. Later, in the initial of step 3 , only disk n is on peg B , and disk1 or disk2 or none is on peg A while others are on peg C ; and in the final state of step 3 , only disk n is on peg B , and disk1 or disk2 or none is on peg C while others are on peg A .Finally ,in the initial state of Step 5, only disk n is on peg C, and disk and disk 1 or disk 2 or none is on peg B while others are on peg A. Hence the transformation diagram is show in Figure-3.3d

Lemma 3.8. For n=2 , the minimum numbers of disk moves for transforming R(A(1, 3,..., n - 1), B(2), C(0)) into R(A(0),B(1,…,n-1), C(0)) and R(A(0), B(0), C(1,..., n)), are c(n)- 3 and a(n)-3 ,respectively .

Proof： We want to prove this lemma by induction.

It is clearly true for n = 2. We assume this lemma is true for n- 1. This assumption implies

(1) the minimal numbers of disk moves for transforming R(A(1, 3,..., n - 1), B(2), C(0)) into R(A(0),B(1,…,n-1), C(0)) and R(A(0),B(0),C(1,…,n-1)), are c(n- 1)- 3 and a(n-1)- 3, respectively;

(2) the minimal numbers of disk moves for transforming R(A(1,...,n- 1), B(0), C(0)) into R((A(2), B(1, 3,...,n 1), C(0)) and R(A(0),B(2), C(1, 3,.,.,n 1)), are c(n-1)-3 and a(n - 1) - 3, respectively.

By Lemmas 3.1 and 3.2 and the above two results,

We obtain

(3) the minimal numbers of disk moves for transforming R(A(2, .,n- 1), B(1), C(0)) into R(A(2), B(1, 3,.., n-1), C(0)) and R(A(1,3,...,n- 1), B(2), C(0)) into R(A(1), B(2,...,n-1), C(0)), are both c(n-1)- 4.

Because the minimal numbers of disk moves for transforming R(A(1,..., n - 1), B(0), C(0)) into R(A(1, 3, '.. ,n 1), B(2), C(0)) and R(A(2),B(1,3,n-1), C(0)) into R(A(0),B(1,2….,n-1), C(0)), are both 3, it is obvious that

(4) the minimal number of disk moves for transforming R(A(1, 3,..., n 1), B(2), C(0)) into R(A(2), B(1, 3,...,n-1), C(0)) is at c(n-1)-6.

By Lemma 3.3 and 3.4 and the above four results, we obtain Figure 3.3a In Figure-12a, there is the optimal transformation of R(A(1,..., n), B(0), C(0)) into R(A(0), B(1,..., n), C(0)):

R (A (1,…, n) , B(0) , C(0))

↓

R (A (n), B (1), C (2, …, n-1))

↓

R (A (2), B (n), C (1, 3,..., n-1))

↓

R (A (0), B (1, …, n) , C(0))

Figure 3.3a Transform R(A(1,...,n), B(0), C(0)) into R（A(0), B(1,2,….,,n), C(0)）

Because the minimal numbers of disks moves for transforming R(A(1,…,n) , B(2) , C(0)) into R(A(1,3,…,n) , B(2) , C(0)) and R(A(1,3,…,n) , B(2) , C(0)))into R(A(n), B(1), C(2,...,n-1)) are 3 and a(n- 1) - 4, respectively, the optimal transformation can be modified as ：

R (A (1,..., n), B(0), C(0))

$\Downarrow$

R (A(1,3,...,n), B(2), C(0))

$\Downarrow$

R (A(n), B(1), C(2,...,n 1)))

$\Downarrow$

R (A(0),B (1,…,n), C(0)).

Figure 3.3b **T**ransform R(A(1,...,n), B(0), C(0)) into R（A(0), B(1,2,….,,n), C(0)）

Hence we have proven that the minimal number of disk moves for transforming R(A(1, 3,..., n),B(2), C(0)) into R(A(0), B(1,... ,n), C(0)) is c(n) -3.

By the same method, we also prove that the minimal number of disk moves for transforming R(A(1, 3,...,n), B(2), C(0)) into R(A(0), B(0),C(1,…,n)) is a(n)-3 in Figure-3.3f

Theorem 3.9 For n=3 ,c(n)=2*a(n-1)-3 , and a(n)=2*a(n-1)+c(n-1)-6

Proof： By Lemma 5 and Figure3.3.5 & Figure 3.3.6 we obtain c(n)=[a(n-1)- 1] + 1 + [a(n-1)- 3] = 2 *a(n-1)-3 and a(n) = [a(n -1)-1] + 1 + [c(n - 1) - 4] + 1 + [a(n-1)-3] = 2 a(n- 1) + c(n- 1)-6,for n =3.

Finally, applying the two recurrence relations in Theorem 6, we obtain c(n) and a(n) as following

Theorem 3.10 For n = 3,

$$\begin{cases} c(n) = [(1+\sqrt{3})^{n-1} + (1-\sqrt{3})^{n-1}]/2 + 3 \\ a(n) = [(1+\sqrt{3})^{n} + (1-\sqrt{3})^{n}]/4 + 3 \end{cases}$$

Proof ：

It is trivial that c(1)=1, a(1)=2, c(2)=4 and a(2)=5 , respectively.

By recurrence relations in Theorem 1, c(n) = 2*a(n-1) - 3 and a(n)=2*a(n-1)+c(n-1)-6, we have

$$\begin{cases} c(n) = [(1+\sqrt{3})^{n-1} + (1-\sqrt{3})^{n-1}]/2 + 3 \\ a(n) = [(1+\sqrt{3})^{n} + (1-\sqrt{3})^{n}]/4 + 3 \end{cases}$$ for n= 3.

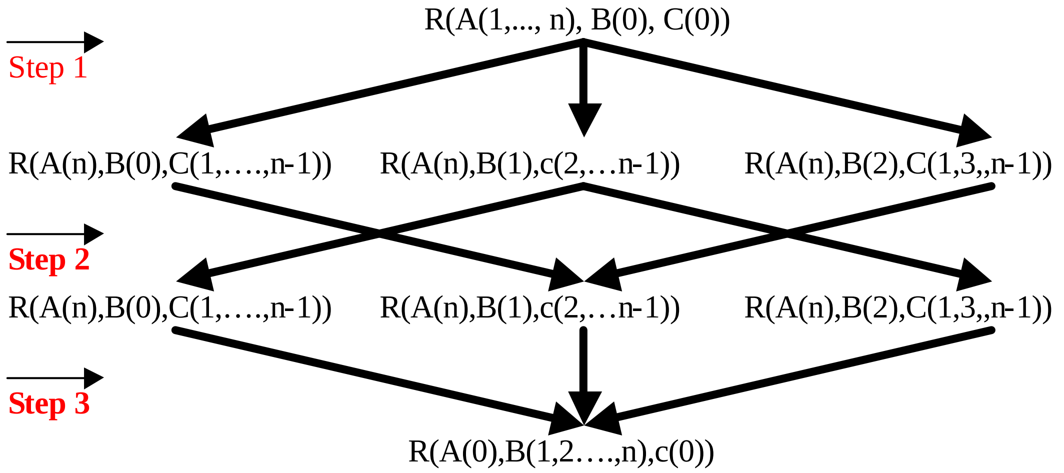


Figure 3.3c Transform R(A(1,...,n), B(0), C(0)) into R（A(0), B(1,2,….,n) C(0)）

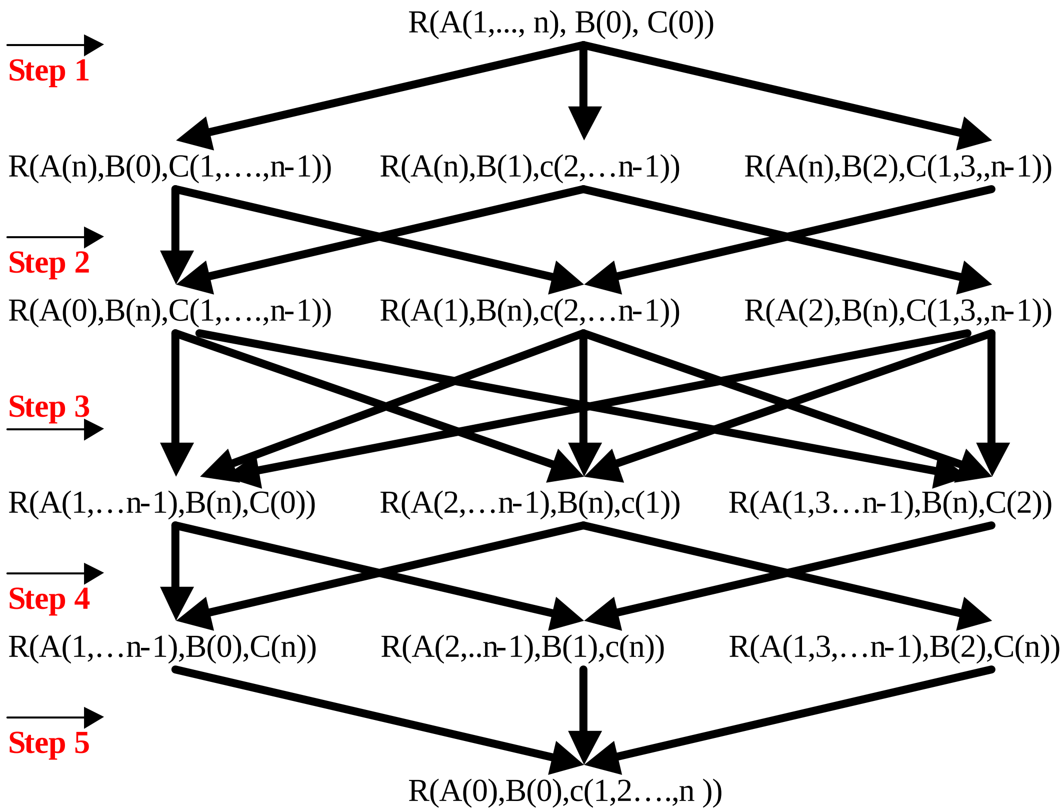


Figure 3.3d Transform R(A(1,...,n), B(0), C(0)) into R(A(0),B(0),C(1,2,….,,n))

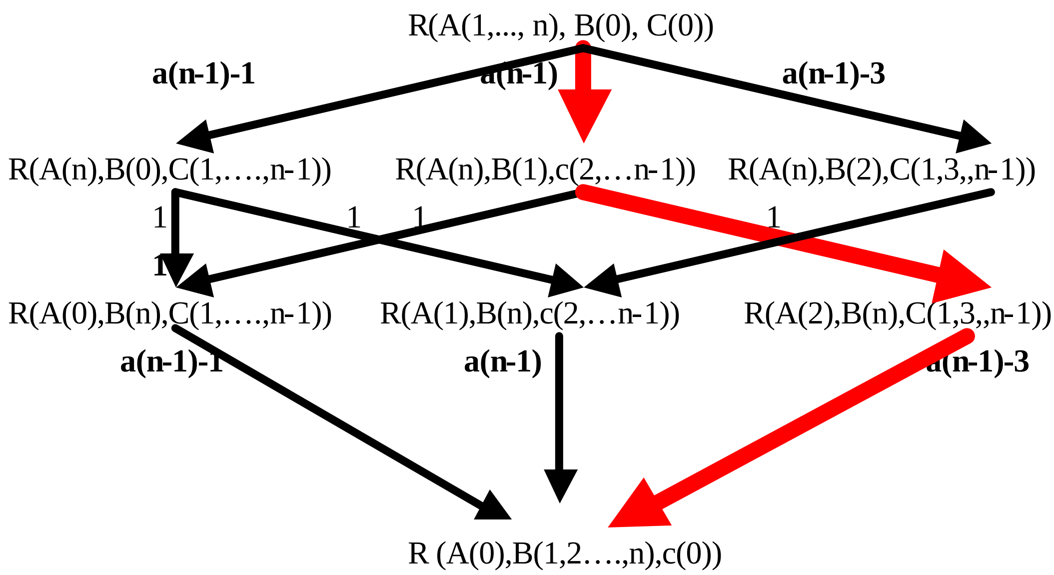


Figure 3.3e Transform R(A(1,...,n), B(0), C(0)) into R（A(0), B(1,2,….,,n), C(0)）

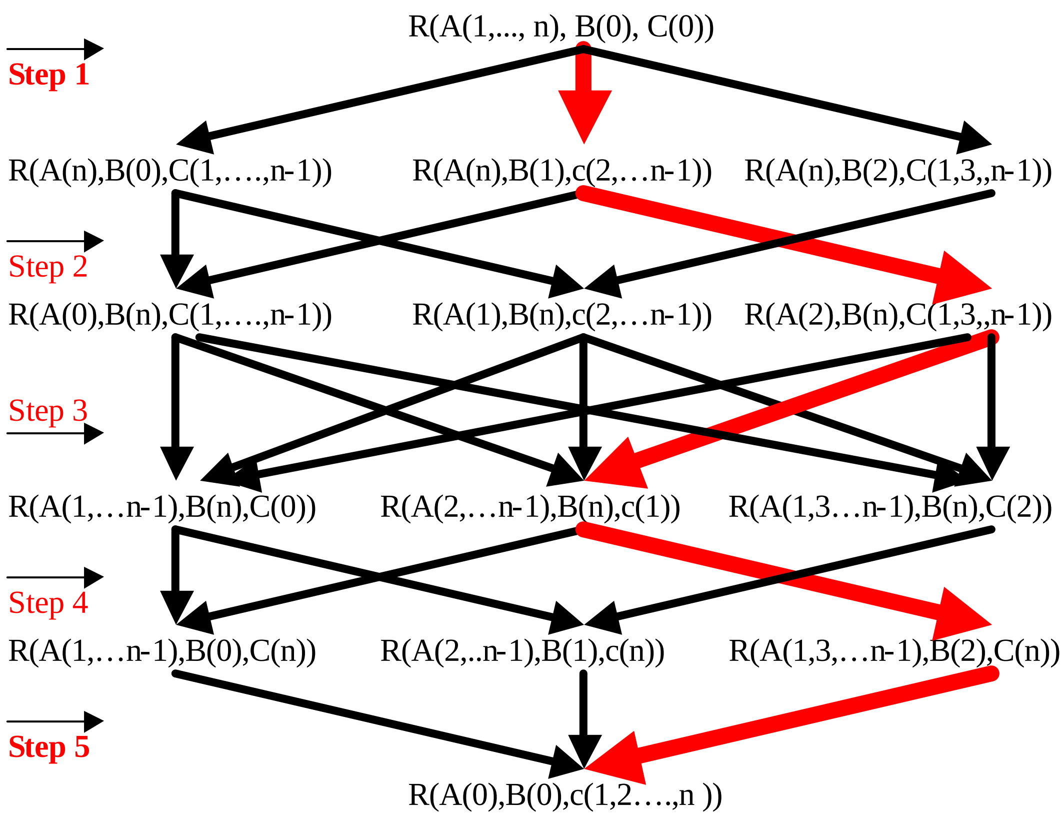


Figure 3.3f Transform R(A(1,...,n), B(0), C(0)) into R（A(0), B(0), C(1,2,….,n))

Chapter 4

Iterative Algorithm on "The Tower of Hanoi Problem"

4.1 Introduction

Like the traditional TTOHP definition in section 2.1 (See Definition1 and Definition2) and rules1, rule2. We will derive an iterative algorithm (Like human work) to solve TTOHP.

4.2 Deriving an Iterative Algorithm for “The Tower of Hanoi problem”

Let's look for a pattern in the number of steps it takes to move just one, two, or three disks. We'll number the disks starting with disk 1 on the top, increasing number to bottom (largest number is at bottom position)

We will see some real case in follow:

Suppose A is source peg, C is destination peg

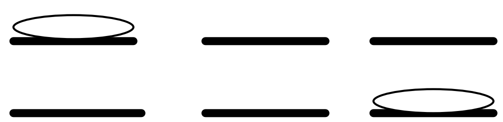
First labeled n disks 1, 2, 3, ……, n from small to large. Disk-1 is the smallest and disk-n is the largest disk .

We also use Figure 4.2a、Figure4.2b、Figure4.2c to illustrate the condition

1 disk: 1 move

Move 1: move disk 1 to peg C

Figure 4.2a One disk move



(anticlockwise)

2 disks: 3 moves

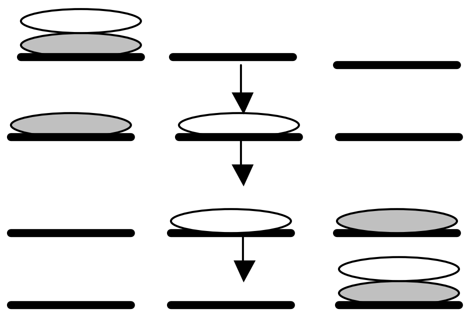
Move 1: move disk 1 to peg B (clockwise)

Move 2: move disk 2 to peg C

(anticlockwise)

Move 3: move disk 1 to peg C (clockwise)

Figure 4.2b Two disks moves



3 disks: 7 moves

Move 1: move disk 1 to peg C

（anticlockwise）

Move 2: move disk 2 to peg B

（clockwise）

Move 3: move disk 1 to peg B

（anticlockwise）

Move 4: move disk 3 to peg C

（anticlockwise）

Move 5: move disk 1to peg A

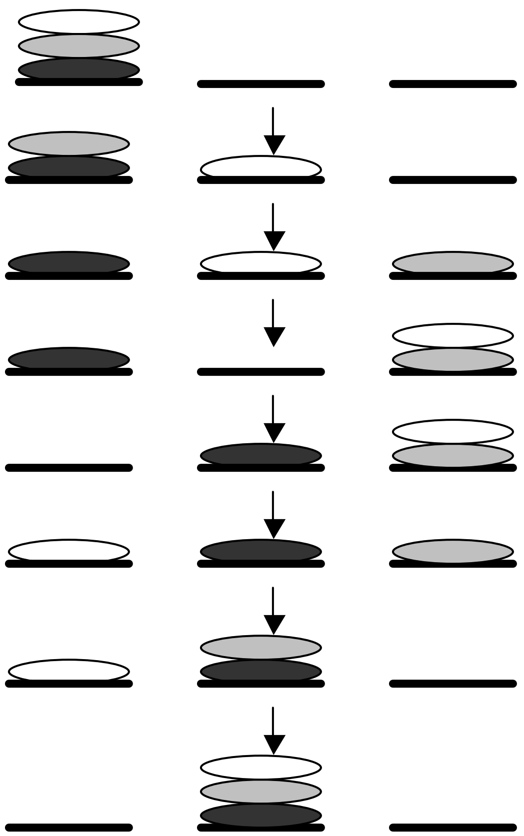
（anticlockwise）

Move 6: move disk 2 to peg C （clockwise）

Move 7: move disk 1 to peg C

（anticlockwise）

Figure 4.2c Three disks moves



From above we observe following properties for f(3,n)

1. If n is odd, first move disk 1 to destination peg , all odd number disk will move by same direction , and all even number disk have opposite direction move. (example：source peg is A and destination peg is B , then first move disk 1 to peg B. All odd number disk are move clockwise direction, all even number move anticlockwise direction.)
2. If n is even, first move disk 1 to spare peg , all odd number disk will move by same direction, and all even number disk have opposite direction move. (example： source peg is A and destination peg is B , then first move disk 1 to peg C, all odd number disk are move anticlockwise direction , all even number move clockwise direction).
3. Next move second smallest disk to next peg by opposite direction .（disk 1 and second smallest disk are move alternative until all disks are on destination peg ）.

4.3 Program and Result

We will show this algorithm by next matlab program

Figure .4.3a is the flowchart of the program

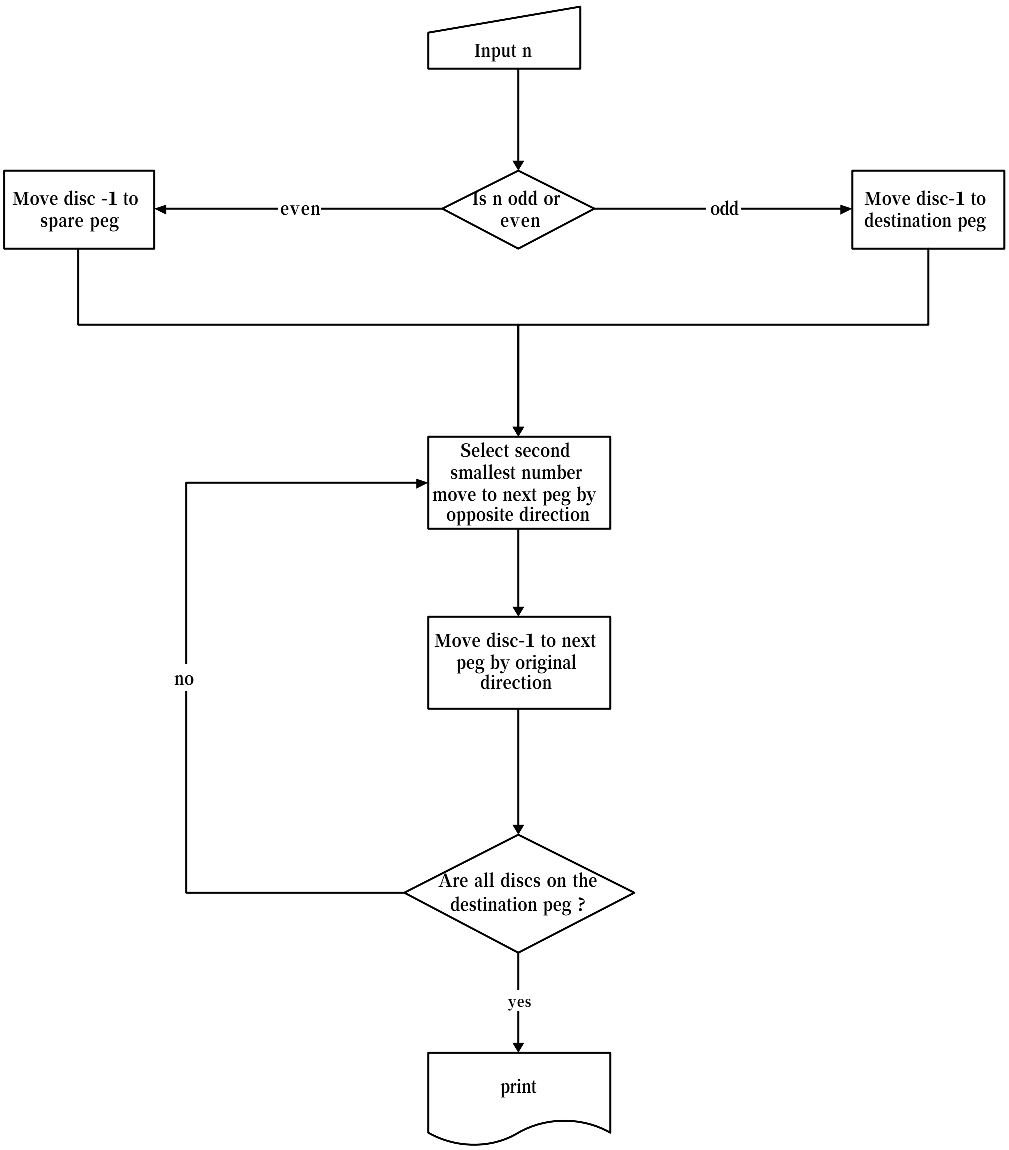


Figure 4.3a Flowchart of TTOHP with iterative algorithm.

<u>Program 1</u>

```
clear all;
anti=1;
clock=0;
n=5;
global n;
A=1:n;
A=A';
B=zeros(n,1);
C=zeros(n,1);
step_of_all=0;
global step_of_all;
[A,B,C];
if mod(n,2)==1,
    while  is_all_disk_in_C(C)==0;

         here=where_is_1(A,B,C);
         there=rotate(anti,here);
         [A,B,C]=putin(1,there,A,B,C);
         [A,B,C]=takeout(1,here,A,B,C);
            [A  B  C]

         [here,second_small]=find_second_small(A,B,C);

        if  mod(second_small,2)==1,
              there=rotate(anti,here);
              [A,B,C]=putin(second_small,there,A,B,C);
              [A,B,C]=takeout(second_small,here,A,B,C);
            [A  B  C]
         else
              there=rotate(clock,here);
              [A,B,C]=putin(second_small,there,A,B,C);
              [A,B,C]=takeout(second_small,here,A,B,C);
            [A  B  C]
         end
```

```
    end
else
    while  is_all_disk_in_C(C)==0,

        here=where_is_1(A,B,C);
        there=rotate(clock,here);
        [A,B,C]=putin(1,there,A,B,C);
        [A,B,C]=takeout(1,here,A,B,C);
           [A  B  C]

        [here,second_small]=find_second_small(A,B,C);

        if  mod(second_small,2)==1,
            there=rotate(clock,here);
            [A,B,C]=putin(second_small,there,A,B,C);
            [A,B,C]=takeout(second_small,here,A,B,C);
           [A  B  C]
        else
            there=rotate(anti,here);
            [A,B,C]=putin(second_small,there,A,B,C);
            [A,B,C]=takeout(second_small,here,A,B,C);
           [A B C]
        end
      end
    end

[A,B,C];
step_of_all
```

We will see two results of real case step by step, like human work**. when n=4 , 5**

In the following figure the number 0 means no disk, smaller number means smaller disk.

A	B	C	A	B	C	A	B	C
original			↓*6*			↓*12*		
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	1	2	0	1	0	3
4	0	0	4	3	0	2	0	4
↓*1*			↓*7*			↓*13*		
0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0
3	0	0	0	2	0	0	0	3
4	1	0	4	3	0	2	1	4
↓2			↓*8*			↓*14*		
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	2
3	0	0	0	2	0	0	0	3
4	1	2	0	3	4	0	1	4
↓*3*			↓*9*			↓*15*		
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	2
3	0	1	0	2	1	0	0	3
4	0	2	0	3	4	0	0	4
↓*4*			↓*10*			Finish		
0	0	0	0	0	0	It match with the recursive solution.		
0	0	0	0	0	0			
0	0	1	0	0	1			
4	3	2	2	3	4			
↓*5*			↓*11*					
0	0	0	0	0	0			
0	0	0	0	0	0			
1	0	0	1	0	0			
4	3	2	2	3	4			

Figure 4.3b The result of TTOHP with iterative algorithm with n=4

original			↓ 5			↓ 10			↓ 15		
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	1	0
3	0	0	1	0	0	0	0	0	0	2	0
4	0	0	4	0	0	2	1	0	0	3	0
5	0	0	5	2	3	5	4	3	5	4	0
↓ 1			↓ 6			↓ 11			↓ 16		
0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	1	0
3	0	0	1	0	0	1	0	0	0	2	0
4	0	0	4	0	2	2	0	0	0	3	0
5	0	1	5	0	3	5	4	3	0	4	5
↓ 2			↓ 7			↓ 12			↓ 17		
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	0	0	0	2	0
4	0	0	4	0	2	2	3	0	0	3	0
5	2	1	5	0	3	5	4	0	1	4	5
↓ 3			↓ 8			↓ 13			↓ 18		
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0
4	1	0	0	0	2	2	3	0	0	3	2
5	2	0	5	4	3	5	4	1	1	4	5
↓ 4			↓ 9			↓ 14			↓ 19		
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	2	0	0	0	1
4	1	0	0	1	2	0	3	0	0	3	2
5	2	3	5	4	3	5	4	1	0	4	5

↓20 0 0 0 0 0 0 0 0 1 0 0 2 3 4 5	↓ 25 0 0 0 0 0 0 0 0 1 2 0 4 3 0 5	↓30 0 0 0 0 0 2 0 0 3 0 0 4 1 0 5
↓21 0 0 0 0 0 0 0 0 0 0 1 2 3 4 5	↓ 26 0 0 0 0 0 0 0 0 1 0 0 4 3 2 5	↓31 0 0 1 0 0 2 0 0 3 0 0 4 0 0 5
↓22 0 0 0 0 0 0 0 0 0 2 1 0 3 4 5	↓ 27 0 0 0 0 0 0 0 0 0 0 1 4 3 2 5	*finish* It match with the recursive solution.
↓23 0 0 0 0 0 0 1 0 0 2 0 0 3 4 5	↓ 28 0 0 0 0 0 0 0 0 3 0 1 4 0 2 5	
↓24 0 0 0 0 0 0 1 0 0 2 0 4 3 0 5	↓ 29 0 0 0 0 0 0 0 0 3 0 0 4 1 2 5	

Figure 4.3c The result of TTOHP with iterative algorithm with n=5

Chapter 5

Iterative Algorithm of cyclic moves

5.1 Introduction

Like the definitions in section 3.1

We have cyclic moves of TTOHP as following rules.

1. The moving direction of a disk must be clockwise.
2. Only the top disk of a tower may be moved at a time.
3. No disk can be placed on a smaller one.

Clockwise: if move direction is A→B→C→ A.

Anticlockwise: if move direction is A→C→B→ A

We will derive an iterative algorithm for cyclic moves of TTOHP.

5.2 Iterative Algorithm of Cyclic Moves

Now we will describe an iterative cyclic move step by step

Algorithm ：

1. for l=n-1 to 1 do
2. First judgment whether n is on peg B (destination peg), if the answer is "yes" then set big= "true" go to 3, else big= "false" go to 4.
3. If big= "true" then judgment whether「ring l is on ring l+1?」, if the answer is yes then set big= "true", else big= "false", and do next l.
4. If big= "false" then judgment whether「ring (l+1)'s peg is following ring l`s peg ?」, if the answer is yes then set big= "true", else big= "false" and do next l.
5. At the last if big= "false" then move disk 1, and repeat step 1 until all disks are on the destination peg.
6. At the last if big= "false" then move second smallest disk , and repeat step 1 until all disks are on the destination peg .

In the following Figure 5.2a is the flowchart.

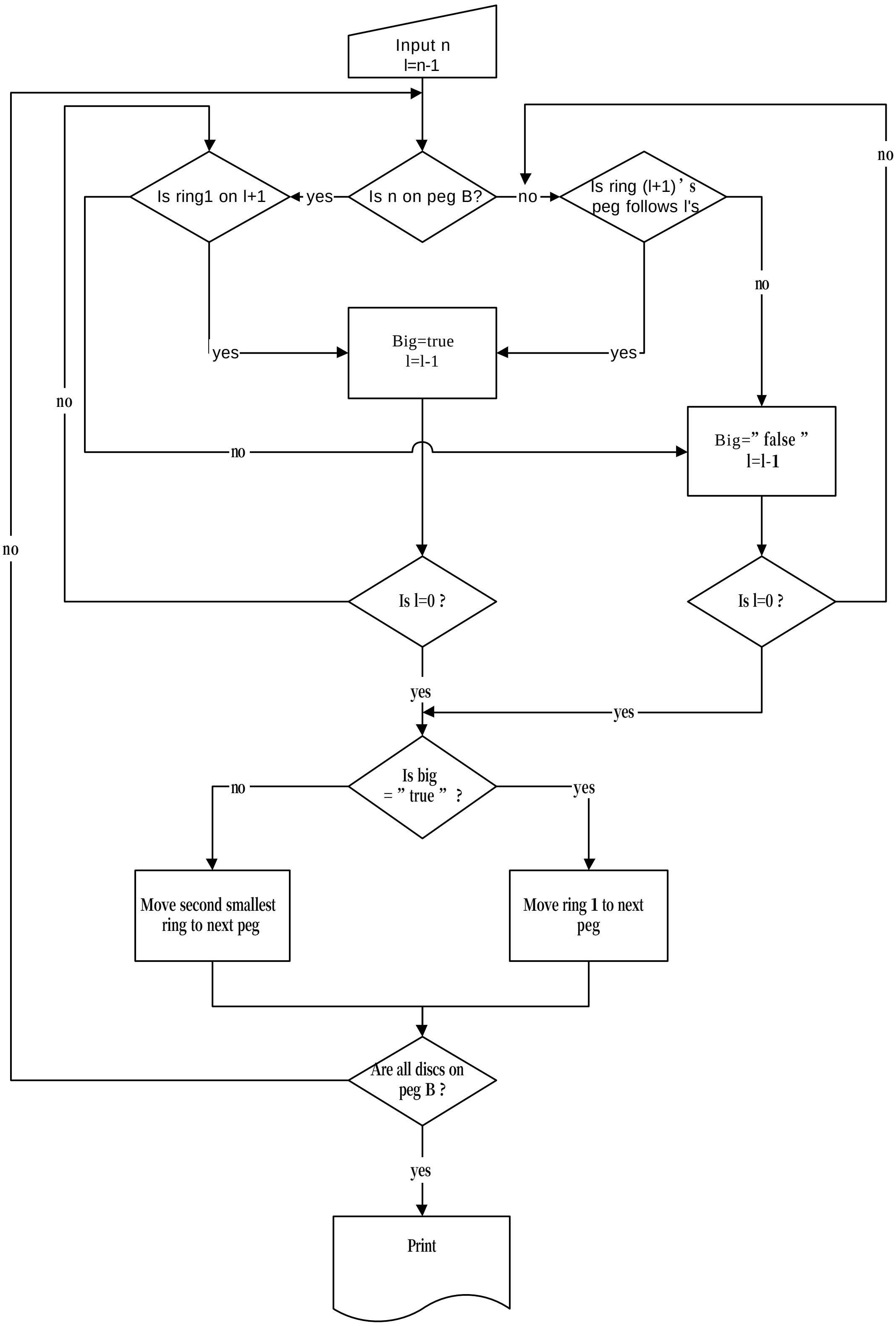


Figure 5.2a Flowchart of the cyclic Tower of Hanoi problem with iterative algorithm.

In the following we will show the iterative method step by step use matlab program

<u>Program 2</u>

```
clear all;
global n;
global step_of_all;
clock=0;
n=4;
big=1;
A=1:n;
A=A';
B=zeros(n,1);
C=zeros(n,1);
step_of_all=0;
[A,B,C]
l=n-1;
all_step=0;

while is_all_disk_in_B(B)==0;
%*********************************************
if is_n_disk_in_B(B)==1;
        if  is_l_on_l1(A,B,C,l)==1,
             big=1;
             l=l-1;
            if  l==0,
                 [A,B,C]=move_disk(A,B,C,big);
             else
                 [big,l]=function_l_on_l1(A,B,C,l);
             end
        else
             big=0;
             l=l-1;
            if  l==0,
                 [A,B,C]=move_disk(A,B,C,big);
             else
                 [big,l]=function_l_isleft_l1(A,B,C,l);
             end
```

```
            end
%=====================================================
               else
%======================================================
==
           if  is_l_isleft_l1(A,B,C,l)==1,
                  big=1;
                  l=l-1;
                 if  l==0,
                         [A,B,C]=move_disk(A,B,C,big);
                  else
                        [big,l]=function_l_on_l1(A,B,C,l);
                  end
            else
                  big=0;
                  l=l-1;
                 if  l==0,
                 [A,B,C]=move_disk(A,B,C,big);
                 else
                        [big,l]=function_l_isleft_l1(A,B,C,l);
                  end
            end
end
%*******************************************************
if l==0,
     [A,B,C]=move_disk(A,B,C,big);
end
l=n-1;
[A,B,C]
all_step=all_step+1;
end
all_step
```

In the next figure we show the real move for step by step f**or n=4**

A	B	C	A	B	C	A	B	C	A	B	C
	original			↓ 6			↓ 12			↓ 18	
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	3	0	0	2	0	0	1	0	0
4	0	0	4	1	2	4	3	1	4	2	3
	↓ 1			↓ 7			↓ 13			↓ 19	
0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0
3	0	0	3	0	1	2	0	0	1	0	2
4	1	0	4	0	2	4	3	0	4	0	3
	↓ 2			↓ 8			↓ 14			↓ 20	
0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	1	2	0	0	0	0	2
4	0	1	4	3	2	4	0	3	4	1	3
	↓ 3			↓ 9			↓ 15			↓ 21	
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	1	0	0	2	0	0	0	0	2
4	2	1	4	3	2	4	1	3	4	0	3
	↓ 4			↓ 10			↓ 16			↓ 22	
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	1	0	2	0	1	0	0	2
4	2	0	4	3	2	4	0	3	0	4	3
	↓ 5			↓ 11			↓ 17			↓ 23	
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	2	1	0	0	0	1	0	0	2
4	0	2	4	3	0	4	2	3	1	4	3

↓ 24 0 0 0 0 0 0 0 1 2 0 4 3	↓ 30 0 0 0 0 1 0 0 2 0 3 4 0	↓ 36 0 0 0 0 0 0 0 3 1 0 4 2	↓ 42 0 0 0 0 2 0 0 3 0 1 4 0
↓ 25 0 0 0 0 0 0 0 1 0 2 4 3	↓ 31 0 0 0 0 0 0 0 2 0 3 4 1	↓ 37 0 0 0 0 0 0 0 3 0 1 4 2	↓ 43 0 1 0 0 2 0 0 3 0 0 4 0
↓ 26 0 0 0 0 0 0 0 0 1 2 4 3	↓ 32 0 0 0 0 0 0 1 2 0 3 4 0	↓ 38 0 0 0 0 1 0 0 3 0 0 4 2	*Finish* All step = 43 It match with the recursive solution.
↓ 27 0 0 0 0 0 0 0 2 1 0 4 3	↓ 33 0 0 0 0 0 0 1 0 0 3 4 2	↓ 39 0 0 0 0 1 0 0 3 0 2 4 0	
↓ 28 0 0 0 0 0 0 0 2 0 1 4 3	↓ 34 0 0 0 0 0 0 0 1 0 3 4 2	↓ 40 0 0 0 0 0 0 0 3 0 2 4 1	
↓ 29 0 0 0 0 1 0 0 2 0 0 4 3	↓ 35 0 0 0 0 0 0 0 0 1 3 4 2	↓ 41 0 0 0 0 2 0 0 3 0 0 4 1	

Figure 5.2b Result of cyclic TTOHP with iterative algorithm with n=4 (clockwise)

For anticlockwise is similarity , we will show in the next

Program 3

```
clear all;
global n;
global step_of_all;
clock=0;
n=4;
big=1;

A=1:n;
A=A';
B=zeros(n,1);
C=zeros(n,1);
step_of_all=0;
[A,B,C]
l=n-1;
all_step=0;
while is_all_disk_in_C(C)==0;
%********************************************
if is_n_disk_in_C(C)==1;

        if  is_l_on_l1(A,B,C,l)==1,
             big=1;
             l=l-1;
            if  l==0,
                 [A,B,C]=move_disk(A,B,C,big);
             else
                 [big,l]=function_l_on_l1(A,B,C,l);
             end
        else
             big=0;
             l=l-1;
            if  l==0,
                 [A,B,C]=move_disk(A,B,C,big);
             else
```

```
                [big,l]=function_l_isleft_l1(A,B,C,l);
            end
        end
%=====================================================
    else
%=====================================================
        if  is_l_isleft_l1(A,B,C,l)==1,
            big=1;
            l=l-1;
           if  l==0,
                [A,B,C]=move_disk(A,B,C,big);
            else
                [big,l]=function_l_on_l1(A,B,C,l);
            end
        else
            big=0;
            l=l-1;
           if  l==0,
                [A,B,C]=move_disk(A,B,C,big);
            else
                [big,l]=function_l_isleft_l1(A,B,C,l);
            end
        end

end
%*****************************************************
if l==0,
    [A,B,C]=move_disk(A,B,C,big);
end

l=n-1;
[A,B,C]
all_step=all_step+1;
end

all_step
```

We will show the result（when n=4）in the following

A	B	C	A	B	C	A	B	C	A	B	C
	original			↓ 6			↓ 12			↓ 18	
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	3	0	0	2	0	0	1	0	0
4	0	0	4	1	2	4	3	1	4	2	3
	↓ 1			↓7			↓ 13			↓ 19	
0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0
3	0	0	3	0	1	2	0	0	1	0	2
4	1	0	4	0	2	4	3	0	4	0	3
	↓ 2			↓ 8			↓ 14			↓ 20	
0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	1	2	0	0	1	0	2
4	0	1	4	3	2	4	0	3	4	0	3
	↓ 3			↓ 9			↓ 15			↓ 21	
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	1	0	0	2	0	0	0	0	2
4	2	1	4	3	2	4	1	3	4	0	3
	↓ 4			↓ 10			↓ 16			↓ 22	
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	1	0	2	0	1	0	0	2
4	2	0	4	3	2	4	0	3	0	4	3
	↓ 5			↓ 11			↓ 17			↓ 23	
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	2	1	0	0	0	1	0	0	2
4	0	2	4	3	0	4	2	3	1	4	3

↓ 24 0 0 0 0 0 0 0 1 2 0 4 3	↓ 30 0 0 0 0 1 0 0 2 0 3 4 0	↓ 36 0 0 0 0 0 0 2 0 0 3 4 1	↓ 42 0 0 0 0 0 0 1 0 0 3 2 4
↓ 25 0 0 0 0 0 0 0 1 0 2 4 3	↓ 31 0 0 0 0 0 0 0 2 0 3 4 1	↓ 37 0 0 0 1 0 0 2 0 0 3 4 0	↓ 43 0 0 0 0 0 0 1 0 2 3 0 4
↓ 26 0 0 0 0 0 0 0 0 1 2 4 3	↓ 32 0 0 0 0 0 0 1 2 0 3 4 0	↓ 38 0 0 0 1 0 0 2 0 0 3 0 4	↓ 44 0 0 0 0 0 0 0 0 2 3 1 4
↓ 27 0 0 0 0 0 0 0 2 1 0 4 3	↓ 33 0 0 0 0 0 0 1 0 0 3 4 2	↓ 39 0 0 0 0 0 0 2 0 0 3 1 4	↓ 45 0 0 0 0 0 1 0 0 2 3 0 4
↓ 28 0 0 0 0 0 0 0 2 0 1 4 3	↓ 34 0 0 0 0 0 0 0 1 0 3 4 2	↓ 40 0 0 0 0 0 0 2 0 1 3 0 4	↓ 46 0 0 0 0 0 1 0 0 2 0 3 4
↓ 29 0 0 0 0 1 0 0 2 0 0 4 3	↓ 35 0 0 0 0 0 0 2 1 0 3 4 0	↓41 0 0 0 0 0 0 0 0 1 3 2 4	↓47 0 0 0 0 0 0 0 0 2 1 3 4

↓ 48 0 0 0 0 0 0 0 1 2 0 3 4	↓ 54 0 0 0 0 0 1 0 0 3 2 0 4	*Finish* All step = 59 It match with the recursive solution
↓ 49 0 0 0 0 0 0 0 1 0 2 3 4	↓ 55 0 0 0 0 0 1 0 0 3 0 2 4	
↓ 50 0 0 0 0 0 0 0 0 1 2 3 4	↓ 56 0 0 0 0 0 0 0 0 3 1 2 4	
↓ 51 0 0 0 0 0 0 1 0 0 2 3 4	↓ 57 0 0 0 0 0 2 0 0 3 1 0 4	
↓ 52 0 0 0 0 0 0 1 0 3 2 0 4	↓ 58 0 0 0 0 0 2 0 0 3 0 1 4	
↓ 53 0 0 0 0 0 0 0 0 3 2 1 4	↓ 59 0 0 1 0 0 2 0 0 3 0 0 4	

Figure 5.2c Iterative algorithm Result for cyclic moves with n=4（Anticlockwise）

Chapter 6

Iterative algorithm of parallel moves

6.1 Introduction

Like [definition 6] of Figure 3.2a four types of parallel moves including (a) single move, (b) exchange, (c) consecutive move, and (d) circular move.

We first describe Walsh's [55,56] iterative algorithm to implement the towers of Hanoi problem without parallel moves as follows.

[Algorithm 1]

Hanoi(n, A, B) //move n disks from peg A to peg B//

{

IF $n \in$ odd, then assign the cyclic order A? B? C? A;

ELSE assign to them the cyclic order A? C? B? A;

(1) Move the smallest *disk* to the next peg in the cyclic order;

(2) **IF** all disks are on the same peg, then **EXIT()**;

ELSE move the second-smallest top *disk* onto the peg

not containing the smallest *disk* and goto (1);

}

The number of disk moves is showed to be $2^n - 1$. Later, we want to modify the above algorithm to implement the towers of Hanoi problem allowing parallel moves.

6.2 Iterative Algorithm of Parallel Moves

Now we describe our iterative algorithm to implement the towers of Hanoi problem with parallel moves. First we define disk groups.

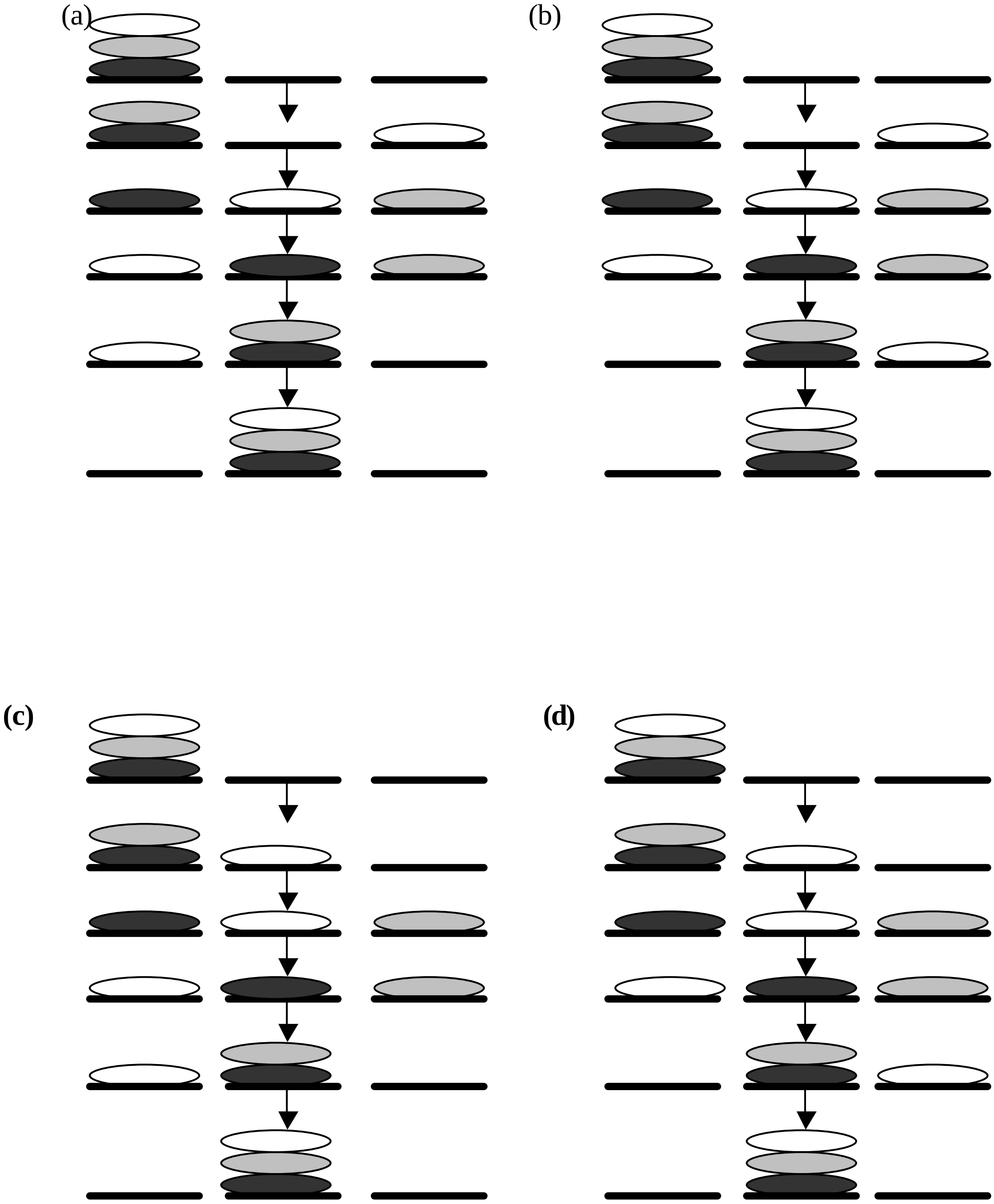
[Definition 7] Disk group

If $n \in$ odd, let disks 1-3 be as group-1, disk 4-5 as group-2, disk 6-7 as group-3, … so there will be (n-1)/2 groups. Otherwise, $n \in$ even, let disks 1-2 be as group-1, disk 3-4 as group-2, disk 5-6 as group-3, …, so there will be n/2 groups.

And then, define all group moves as follows.

[Definition 8] Group-1 move for disks 1-3: 5 parallel moves

Consider source peg is A, (1) destination peg is B or C, (2) the first move of disk 1 to peg B or C, and (3) the last move of disk 1 from non-destination two pegs, so there are total 8 types of group-1 move for disks 1-3 with 5 parallel moves.


(a)
(b)
(c)
(d)

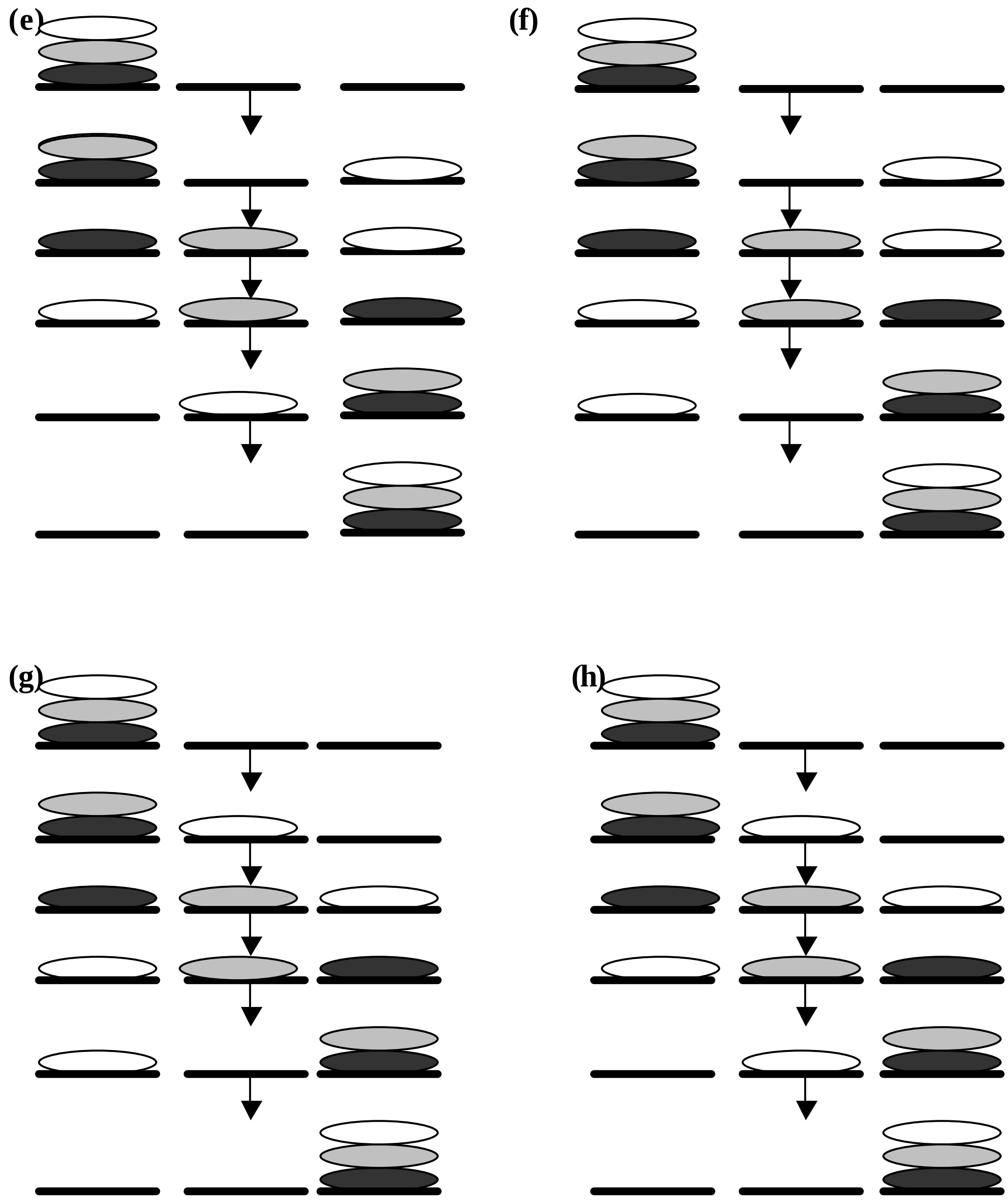


Figure 6.2a Total 8 types of group-1 move for disks 1-3 with 5 parallel moves.

[Definition 9] Group-1 move for disks 1-2 : 3 parallel moves

Consider source peg is A, (1) destination peg is B or C, (2) the first move of disk 1 to peg B or C, and (3) the last move of disk 1 from non-destination 2 pegs, so there are total 8 types of group-1 move for disks 1-2 with 3 parallel moves.

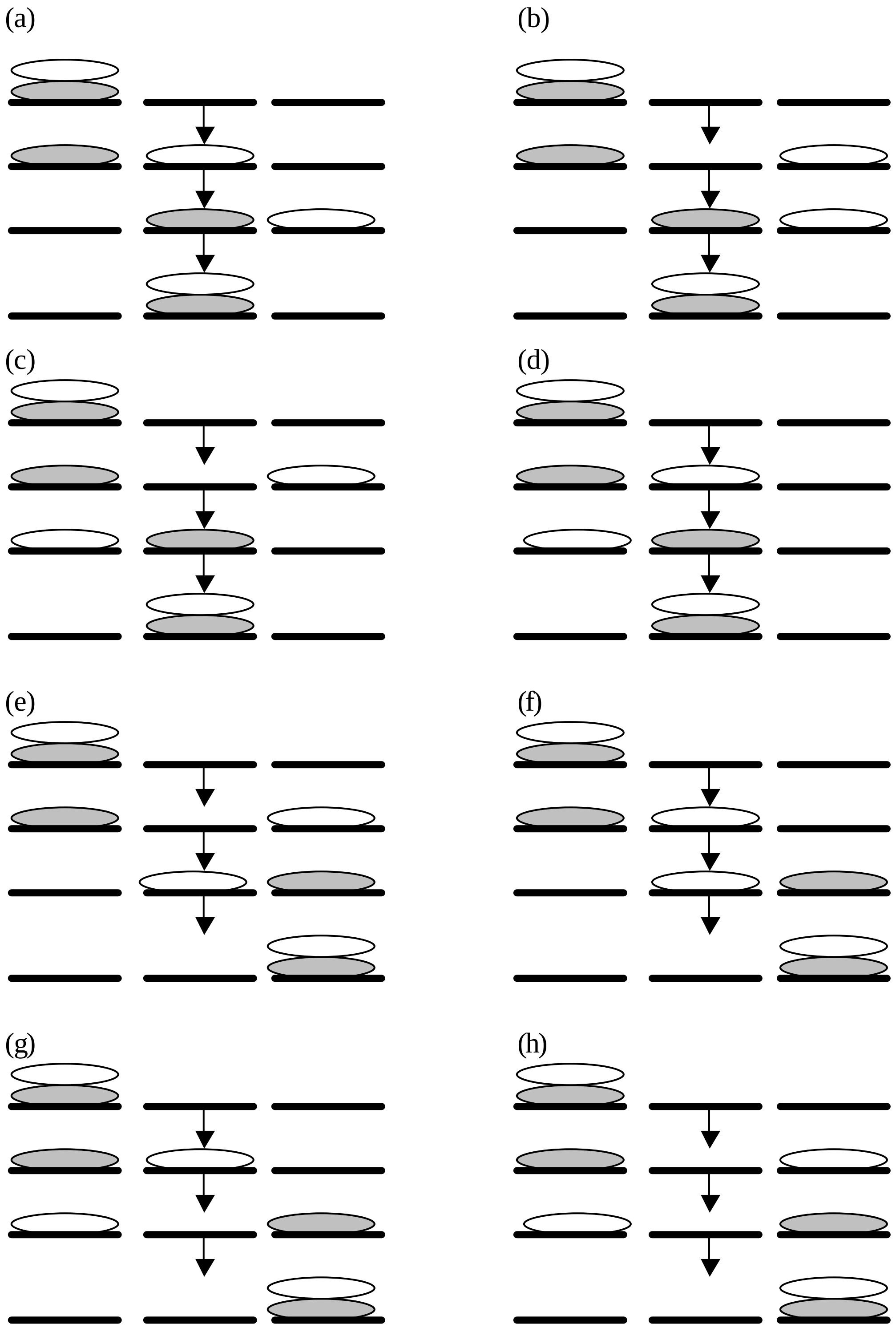


Figure 6.2b Total 8 types of group-1 move for disks 1-2 with 3 parallel moves.

[Definition 10] Other group move for disks i-(i+1) : 3 parallel moves

Consider source peg is A, (1) destination peg is B or C, so there are two types of other group moves with 3 parallel moves.

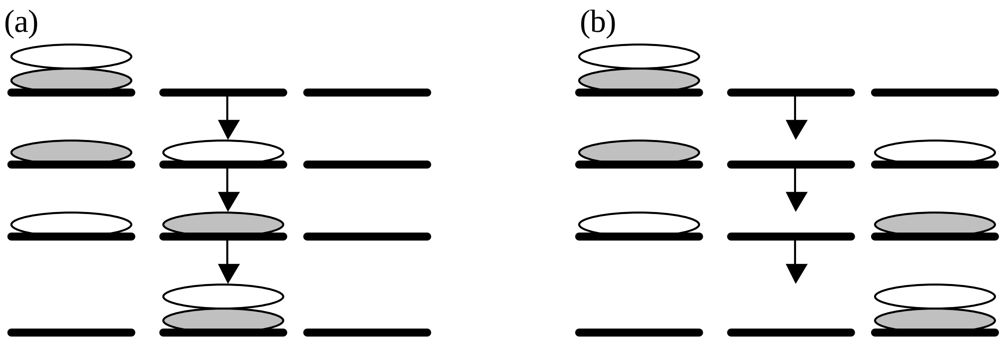


Figure 6.2c Two types of other group moves with 3 parallel moves.

Now we present our optimal parallel iterative algorithm as follows.

[Algorithm 2]

Parallel_Hanoi(n, A, B) //move n disks from peg A to peg B//

{

IF $\lfloor n/2 \rfloor$ *group* $\in$ odd, then assign the cyclic order A? B? C? A;

// n $\in$ odd, there are (n-1)/2 groups; n $\in$ even, there are n/2 groups//

ELSE assign to them the cyclic order A? C? B? A;

(1) Move* the smallest *group* to the next peg in the cyclic order;

(2) **IF** all *groups* are on the same peg, then **EXIT()**;

ELSE move** the second-smallest top *group* onto the peg not containing the smallest *group* and goto (1);

}

* : First selecting group-1 move among 8 types : the first move of disk 1 to the peg not containing the above second-smallest top group, and the last move of disk 1 from the peg not containing the next second-smallest top group. And then, we can combine the first move of disk 1 with the last move of the above second-smallest top group move, and the last move of disk 1 with the first move of the next second-smallest top group move.

**: Combine the first move with the last move of the above smallest group move, and the last move with the first move of the next smallest group move.

[Theorem 6.1] The above iterative parallel algorithm is optimal.

Proof: Consider Algorithm 1 with n disks, there are total $2^n - 1$ disk moves; 2^{n-1} for the smallest disk, and $2^{n-1} - 1$ for the second-smallest disk. Now consider Algorithm 2 with n disks allowing parallel moves.

If $n \in$ odd, there are $(n-1)/2$ groups with $2^{(n-1)/2} - 1$ group moves; where $2^{(n-1)/2-1}$ for the group-1, and $2^{(n-1)/2-1} - 1$ for the other group. Without combination, there are total $5 \cdot 2^{(n-1)/2-1} + 3 \cdot (2^{(n-1)/2-1} - 1) = 4 \cdot 2^{(n-1)/2} - 3$ parallel disk moves.

Applying combination moves: * and **, because there are total $2^{(n-1)/2} - 1$ group moves, the numbers of parallel disk move are

$$[4 \cdot 2^{(n-1)/2} - 3] - [2^{(n-1)/2} - 1 - 1] = 3 \cdot 2^{(n-1)/2} - 1.$$

If $n \in$ even, we similarly derive the number, without combination there are total $3 \cdot 2^{n/2-1} + 3 \cdot (2^{n/2-1} - 1) = 3 \cdot 2^{n/2} - 3$ parallel disk moves. Applying combination moves, all the numbers of parallel disk move are

$$[3 \cdot 2^{n/2} - 3] - [2^{n/2} - 1 - 1] = 2 \cdot 2^{n/2} - 1.$$

The above result is the same as the optimal disk move number derived by Wu & Chen[57], so our algorithm is the optimal iterative parallel.

Chapter 7

Conclusion

7.1 Result of Our Research

In this paper, we study recursive and iterative algorithms on “The tower of Hanoi problem” with three pegs. Six surveys are introduce : four recursive algorithm, (1) traditional TTOHP, (2) cyclic moves TTOHP, (3) parallel moves TTOHP, (4) cyclic parallel moves TTOHP, and two iterative algorithm (5) iterative algorithm on traditional TTOHP (6) iterative algorithm on cyclic TTOHP. In chapter 6 we introduce our observation: design a simple and effective iterative algorithm to implement TTOHP with parallel moves.

We propose the algorithm is base on the theorem of Walsh’s thesis: the iterative optimal solutions for TTOHP with cyclic moves. Add the definition of

WU&CHEN in the parallel moves of TTOHP, we design the simple algorithm, and have proved its solution is same with the solution of recursive algorithm.

In the finally we list some data for TTOHP to compare.

TTOHP SOLUTION						
algorithm	standard	cyclic		parallel	cyclic parallel	
		c(n)	a(n)		c(n)	a(n)
Formula / Disk number	2^n-1	$2a_{n-1}+1$	$2a_{n-1}+c_{n-1}+2$		$2a_{n-1}-3$	$2a_{n-1}+c_{n-1}-6$
1	1	1	2	1	1	2
2	3	5	7	3	4	5
3	7	15	21	5	7	8
4	15	43	59	7	13	17
5	31	119	163	11	31	41
6	63	327	447	15	79	107
7	127	895	1223	23	211	287
8	255	2447	3343	31	571	779
9	511	6687	9135	47	1555	2123
10	1023	18271	24959	63	4243	5795
11	2047	49919	68191	95	11587	15827
12	4095	136383	186303	127	31651	43235
13	8191	372607	508991	191	86467	118115
14	16383	1017983	1390591	255	236227	322691
15	32767	2781183	3799167	383	645379	881603
16	65535	7598335	10379519	511	1763203	2408579

Figure 7.1 Some optimal solution for TTOHP and its variants

7.2 Studying Topic in the Future

The multi-peg (k≧4) TTOHP is another generalized version of this classical problem, and is still open. Newman-Wolfe [49] obtained some lower and upper bounds on the number of moves for different ranges of the number of pegs. They had given a recursive formulation for computing the number of moves. But interestingly, it is not known whether the number of moves given in Boardman's formulation is optimal and no evidence to contrary is available either. In the finally, we provide this topic for the future research on TTOHP .

Reference

[1] J. P. Allouche, "Note on the Cyclic Towers of Hanoi", TCS 123 (1), pp.3-7, 1994.

[2] M. D. Atkinson, "The cyclic towers of Hanoi", Information Processing Letters 13, pp.118-119, 1981.

[3] P. Buneman & L. Levy, "The towers of Hanoi problem", Information Processing Letters 10, pp.243-244, 1980.

[4] W. C. Chang & G. J. Chang, "Study on Tower of Hanoi", Master Thesis, Submitted to Department of Applied Mathematics, National Chiao Tung University Engineering, 1998.

[5] F. B. Chedidand Toshiya Mogi, "A simple iterative algorithm for The Tower of Hanoi Problem", IEEE Trans. Ed. 39, no. 2, pp. 274-275, 1996.

[6] E. W. Dijkstra, "A short introduction to the art of programming", EWD 316, 1971.

[7] B. Eggers, “The towers of Hanoi: Yet another nonrecursive solution”, SIG-PLAN Notices 20, no. 9 (September), pp.32-42, 1985.

[8] M. C. Er, “A general algorithm for finding a shortest path between two n-configurations”, Inform. Sci. 42, pp.137-141, 1987.

[9] M. C. Er, “A generalization of the cyclic towers of Hanoi: An iterative solution”, Internat. J. Comput. Math. 15, pp.129-140, 1984.

[10] M. C. Er, “A loop less approach for constructing a fastest algorithm for the Towers of Hanoi problem”, Internat. J. Comput. Math. 20, pp.49-54, 1986.

[11] M. C. Er, “A loop less and optimal algorithm for the cyclic towers of Hanoi problem”, Inform. Sci. 42, pp.283-287, 1987.

[12] M. C. Er, “A minimal space algorithm for solving the towers of Hanoi problem”, J. Inform. Optim. Sci 9, pp.183-191, 1988.

[13] M. C. Er, “A representation approach to the tower of Hanoi problem”, Comput. J. 25, no. 4, pp.442-447, 1982.

[14] M. C. Er, “An algorithmic solution to the multi-tower of Hanoi problem”, J. Inform. Optim. Sci. 8, no. 1, pp.91-100, 1987.

[15] M. C. Er, “An analysis of the Generalized Tower of Hanoi Problem”, Bit 23 (4), pp. 429-435, 1983.

[16] M. C. Er, “An iterative algorithm for the cycle towers of Hanoi problem”, Internat. J. Comput. Inform. Sci. 13, no. 2, pp.123-129, 1984.

[17] M. C. Er, “An iterative solution to the Generalized Tower of Hanoi Problem”, Bit 23 (4), pp.295-302, 1983.

[18] M. C. Er, “An optimal algorithm for Revel’s puzzle”, Inform. Sci 45, pp.39-49, 1988.

[19] M. C. Er, “Counter examples to adjudicating a tower of Hanoi contest”, Internat. J. Comput. Math. 21, pp.123-131, 1987.

[20] M. C. Er, “Performance evaluations of recursive and iterative algorithms for the towers of Hanoi problem”, Computing 37, PP.93-102, 1986.

[21] M. C. Er, “The colour towers of Hanoi-An iterative solution”, J. Inform. Optim Sci. 5, no. 2, pp.95-104, 1984.

[22] M. C. Er, “The complexity of the generalized cyclic towers of Hanoi problem”, J. Algorithms 6, pp.351-358, 1985.

[23] M. C. Er, “The Cyclic Tower of Hanoi: A representation Approach”, The Computer Journal 27 (2), pp.171-175, 1984.

[24] M. C. Er, “The cyclic towers of Hanoi and pseudo ternary code”, J. Inform. Optim. Sci. 7, no. 3, pp.271-277, 1986.

[25] M. C. Er, “The Generalized colour Tower of Hanoi Problem： An iterative algorithm”, The Computer Journal 27 (3), pp.278-282, 1984.

[26] M. C. Er, “The generalized towers of Hanoi problem”, J. Inform. Optim. Sci 5, no. 1, pp.89-94, 1984.

[27] M. C. Er, “The tower of Hanoi problem-a further reply”, Comput. J. 28, no.5, pp.543-544, 1985.

[28] M. C. Er, “The towers of Hanoi and binary numerals”, J. Inform. Optim. Sci. 6, no. 2, pp.147-152, 1985.

[29] M. C. Er, “Towers of Hanoi with black and white disks”, J. Inform. Optim. Sci. 6, no. 1, pp.87-94, 1985.

[30] D. Gault & M. Clint: “A Fast Algorithm for the Towers of Hanoi”, The Computer Journal 30 (4), pp.376-378, 1987.

[31] T. Gedeon, The “Cyclic Tower of Hanoi: An Iterative Solution Produced by Transformation”, The Computer Journal 39 (4), pp.353-356, 1996.

[32] T. D. Gedeon, “The Reve`s puzzle：An iterative solution produced by transformation”, The Computer Journal 35 (2), pp.186-187, 1992.

[33] P. J. Hays, “A note on The Tower of Hanoi problem”, Computer Journal 20, pp. 282-285, 1977.

[34] A. M. Hinz, "An iterative algorithm for the Tower of Hanoi with four pegs", Computing 42, pp.133-140, 1989.

[35] X. –M Lu & T. S. Dillon, "Nonrecursive solution to parallel multipeg Towers of Hanoi: A decomposition approach", Math. Comput. Modelling 24, no. 3, pp.29-35, 1996.

[36] W. Lunnon, "The Reve's Puzzle", Comput. J. 29, p.478, 1986.

[37] A. A. K. Majumdar, "A note on the iterative algorithm for the Reve's puzzle", The Computer Journal. 37（5）pp.463-464, 1994.

[38] A. A. K. Majumdar & M. Kaykobad, "An iterative algorithm for the 5-peg tower of Hanoi problem", J. Bangladesh Acad. Sci. 20, no. 2, pp.119-128, 1996.

[39] A. A. K. Majumdar, "A note on the generalized multi-peg tower of Hanoi problem", Prpc. Pakistan Acad. Sci. 33, no. 1-2, pp.129-130, 1996.

[40] A. A. K. Majumdar, "A note on the iterative algorithm for the four peg tower of Hanoi problem", Bangladesh Acad. Sci. 18, no2, pp.241-250, 1994

[41] A. A. K. Majumdar, "Frame's conjecture and the tower of Hanoi problem with four pegs", Indian J. Math. 36, no. 3, pp. 215-227, 1994

[42] A. A. K. Majumdar, "Generalized multi-peg tower of Hanoi problem", J.

Austral. Math. Soc. Ser. B 38, Q.O. 2, pp 201-208, 1996.

[43] A. A. K. Majumdar, "The divide-and-conquer approach to the generalized p-peg tower of Hanoi problem", Optimization 34, pp.373-378, 1995.

[44] A. A. K. Majumdar, A note on the cyclic towers of Hanoi, Proc. Pakistan A ad. Sci. 33, no. 1-2, pp.131-132, 1996.

[45] A. A. K. Majumdar, "The generalized four-peg tower of Hanoi problem", Optimization, vol 29, pp.349-360, 1994.

[46] A. A. K. Majumdar, "The generalized p-peg tower of Hanoi problem", Optimization, vol 32, pp 175-183, 1995.

[47] S. Margarita, "The towers of Hanoi: a new approach", AI Expert 8,no. 3 (March), pp.22-27, 1993.

[48] S. Minsker, "The Towers of Antwerpen problem", Information Processing Letters 38, pp.107-111, 1991.

[49] R. Newman-Wolfe, "Observations on multipeg Tower of Hanoi", TR187, University of Rochester, 1986.

[50] A. Pettorossi, "Tower of Hanoi Problem: Deriving Iterative Solutions by Program Transformations", Bit 25, pp.327-334, 1985.

[51] J. S. Rohl, "Tower of Hanoi: The Derivation of Some Iterative Versions", The

Computer Journal 30 (1), pp.70-76, 1987.

[52] U. k. Sakar, “On the design of a constructive algorithm to solve the multi-peg towers of Hanoi problem”, Theoretical Computer Science 237, pp.407-421, 2000.

[53] B. M. Stewart, “Solution to 3918”, Amer. Math. Monthly 48, pp.217-219, 1941.

[54] P. k. Stockmeyer, “The Tower of Hanoi: A History Survey and Bibliography”, Department of Computer Science College of William and Mary, 2001.

[55] T. R. Walsh, “Iteration strikes back – at the cyclic towers of Hanoi”, Information Processing Letters 16, pp.91-93, 1983.

[56] T. R. Walsh, “The towers of Hanoi revisited: moving the rings by counting the moves”, Information Processing Letters 15, pp.64-67, 1982.

[57] J.-S. Wu & R. -J. Chen, “The towers of Hanoi problem with parallel moves”, Information Processing Letters 44, pp.241-243, 1992.

[58] J.-S. Wu & R. -J. Chen, “The towers of Hanoi problem with cyclic parallel moves”, Information Processing Letters 46, pp.1-6, 1993.

[59] R .L. Wu & R. J. Chen, “Parallel Tower of Hanoi”, Master Thesis, Submitted to Department of Computer Science and Information Engineering, National Chiao Tung University, 1999.