臺灣二 OO 四年國際科學展覽會

科　　別：電腦科學科

作品名稱：將錯就錯的 knuth 河內塔

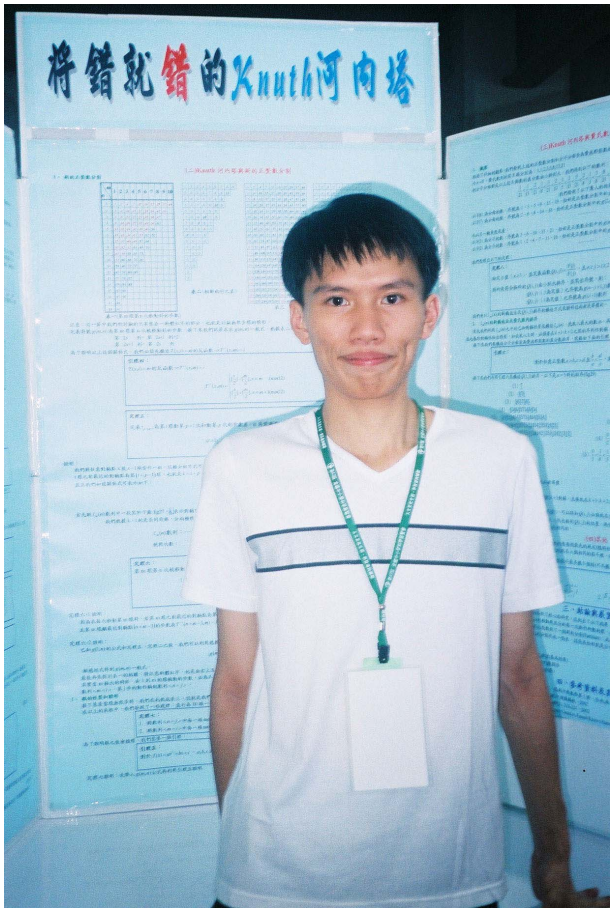
得獎獎項：電腦科學科第一名
美國第五十五屆國際科技展覽會團隊正選代表
英特爾電腦科學獎第一名

學　　校：臺北市立建國高級中學

作　　者：余相甫、李安盛

作 者 簡 介

我的名字叫做余相甫，從小受到父親的影響，我就喜歡獨立思考問題，直到進了建中之後，遇到一了位「教官」，他真正的帶領了我們做了真正的研究，體會到了什麼叫做數學之美。這次的研究除了這位「教官」之外，我想我還要感謝我的雙親，還有一直協助我們的老師(孫老師、文老師、李組長、沈組長、多位的實習老師等)，還有許多幫助我們的同學(坤興、硯仁、Jason 等)，最後就是我最重要的夥伴—安盛，如果我能因此有一點點的榮耀，那麼將歸於所有幫助過我的人。



我的名字叫做李安盛，在今年暑假之前都還是建中的學生。我的興趣是閱讀和集郵。我從小就對各種學問有所涉獵，尤其是數學方面，常常參加數學科展還有小型數學競賽，另外感謝我的父母，因爲完全都不管我，我才能從事自己有興趣的學科。另外，還要感謝高中的數學老師，不僅僅修正了我在各方面的求學觀念，也帶了我們做出這一次的科展作品。最後，我希望在高中三年後，還能時時回想起這一次的轟轟烈烈。

將錯就錯的 Knuth 河內塔

摘要

在這篇報告中，我們探索了「將錯就錯的 Knuth 河內塔問題」。

傳統河內塔問題在電腦科學上佔有重要的地位，是一個極具內涵的模型。由於這個模型的深厚數學內涵，使其和巴斯卡三角形建立了緊密的連結，且利用這個緊密的數學連結，設計出復原任意起始狀態的良好演算法。

Knuth 河內塔起因於數學家 Knuth 在論文[3]中，描述傳統的河內塔問題時所發生的一次筆誤。在這個新的規則之下，我們意外發現 Knuth 河內塔存在著一個和傳統河內塔平行的模型，此模型在電腦科學及數學上有著完全不同於傳統河內塔的內涵。

我們的研究主要如下：(分別為內文中的四大段)

(一) 結構分析。移動環所需要的次數，如何移動環並分析每一次動作所動的環，及每個環何時被動到並給出演算法。

(二) 正整數的分割。所有的移動步驟將正整數做了一個新的分割(Partition)；此分割模 k 之後有良好的循環性質。

(三) 費波那契真分數的排序。這個正整數的分割形成一張表，這張表恰好就是分子分母皆為費波那契真分數之排序。

(四) 隨意亂排的 Knuth 河內塔復原演算法。在 Knuth 河內塔的規定下將起始狀態改變，找出良好的復原演算法，並分析。

Abstract

*In this project we study the **"Knuth Hanoi Tower"**, which is motivated by a typo in a paper of Knuth. This inadvertently typo leads to a new rule of moving the discs on the Hanoi Tower (see introduction below for definition).*

*Although seemingly similar to the traditional Hanoi-Tower problem, it turns out that under this rule the **"Knuth Hanoi Tower"** problem consists of amazing properties, and is totally different from the traditional one.*

Our study focuses on the following directions:

***(1) Structure analyzing:** We analysis the sequences recording the disc moving and offer enumeration results and recurrsive/non-recurrsive algorithms.*

***(2) Partition of N:** The moving sequence forms a partition (a table) of N, which has an amazing congruence property.*

***(3) The order of Fibonacci proper fraction:** The row/column of the partition table is, even more amazing, exactly the order when sorting the Fibonacci proper fraction with fixed denominator/numerator.*

***(4) The Restoration of an arbitrary initial state:** We offer an efficient algorithm for restoring any initial state of discs.*

*We hope that our study on the **"Knuth Hanoi Tower"** offers a simple, neat, and new example on **the theory of Algorithm, Number theory and Combinatorics**.*

一、前言

在最近的一篇論文[3]中，Stanford 大學著名的數學家 Donald E. Knuth 以如下的敘述說明我們所熟知的河內塔問題：

> *「Do people know the **'Tower of Hanoi's problem'**? You have 3 pegs, and you have disks of different sizes. You're supposed to transfer the disk from one peg to another, and the disks have to be sorted so that **the biggest** is always at the **bottom**. You can move only disk at a time.」*
>
> 「你們知道河內塔問題嗎？你有 3 根柱子和一些不同尺寸的環，你應該要將這些環從一根柱子移到另一根柱子上，而且這些環必須要把**最大**的環排在最下面。你一次只能動一個環。」

這顯然是 Knuth 的筆誤。因為原本河內塔的規則是，在任何地方、任何時候，**較大**的環都不能在較小的環上面。這個筆誤被 Stanford 大學的 J. McCarthy 教授發現，他問了一個有趣的問題：

「將錯就錯按照 Knuth 的規則，河內塔問題會變得如何？」

這篇報告就是對於這個將錯就錯的 Knuth 河內塔問題做一個全面的探討。

二、研究過程及方法

（零）Knuth 河內塔

我們在此重新敘述 Knuth 河內塔問題：一開始有 3 根柱子和一些不同尺寸的環，這些環由小到大、由上到下排列在一根柱子上，現在要將這些環全部移到另一根柱子並由小到大、由上到下排列。每次只能移動一個環，但是在移動的過程中，每根柱子上**最大**的環必須在最下面。在這個規則之下，河內塔問題會變得如何？

我們首先舉一個 5 個環的例子（$m=5$）。在這裡框代表柱子，數字編號代表環，編號越大代表環的直徑越大。

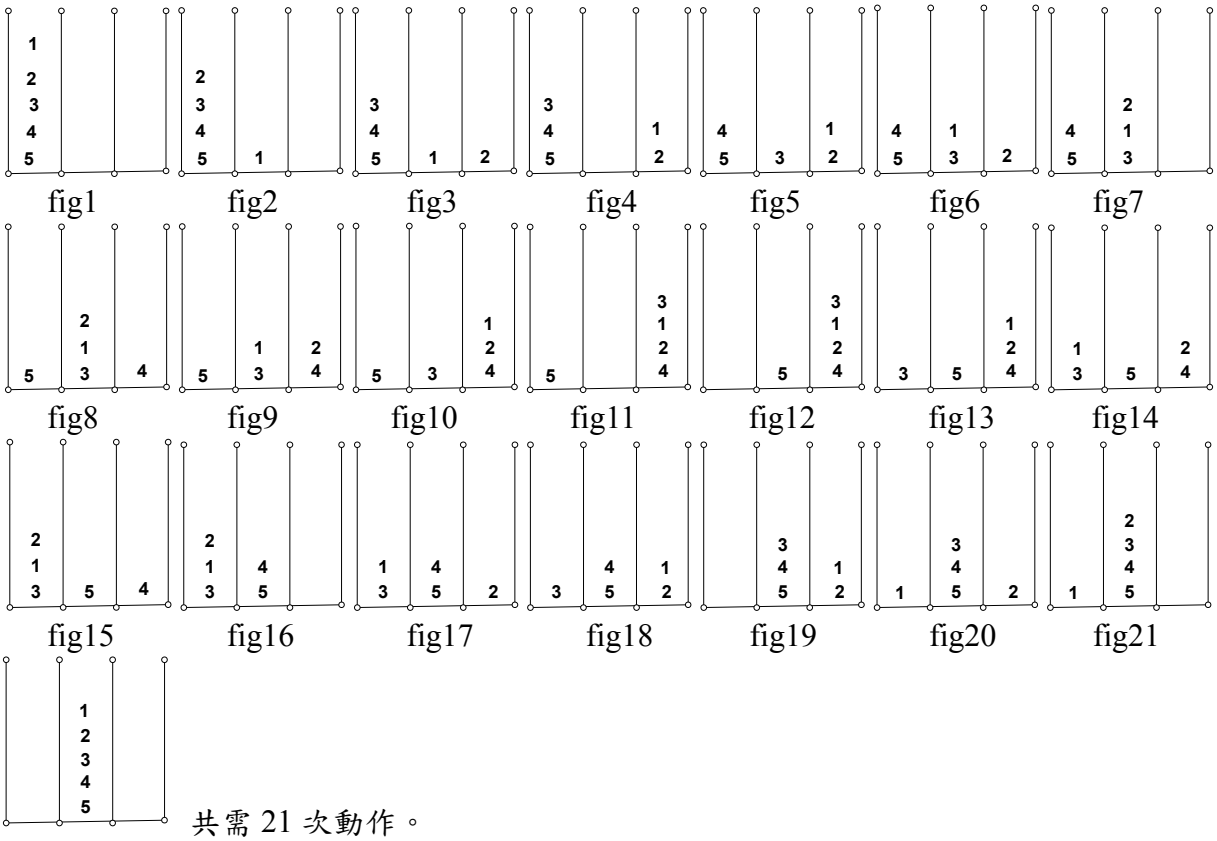


fig1 fig2 fig3 fig4 fig5 fig6 fig7

fig8 fig9 fig10 fig11 fig12 fig13 fig14

fig15 fig16 fig17 fig18 fig19 fig20 fig21

fig22

共需 21 次動作。

我們要在此強調 Knuth 河內塔規則和傳統的河內塔規則非常相似，但完全不同。按照 Knuth 河內塔規則，它只需要 21 次動作，而在原本河內塔規則中卻需要 $2^5-1=31$ 次動作。

以下我們分四節來討論：

(一) Knuth 河內塔結構分析

(二) 正整數的分割

(三) 費波那契真分數的排序

(四) 隨意亂排的 Knuth 河內塔復原演算法

(一)　Knuth 河內塔結構分析

首先我們針對 **Knuth 河內塔**的操作過程進行分析，我們希望藉由整個結構分析幫助我們了解 **Knuth 河內塔**的內涵以助於我們設計演算法。

1、 移動所有環所需要的次數

一開始我們如同原來的河內塔一樣，先計算移動所有環所需要的次數，我們的策略是觀察移動環的過程，找出一般的規律，並給出遞迴式計算移動所有環所需要的次數。

觀察第零節五個環的例子，會發現：在動第 5 環之前，為了動某個環，必須將所有小於欲動的環，移到其中一根柱子上，例如：第零節中 fig5 到 fig7，為了動第 4 環，必須將第 1、2、3 環全部移到一根柱子上。另外在第零節中 fig11 到 fig12，最大的環移動之後，為了將所有的環再依序排到最大的環上面，只須逆操作 fig1 到 fig11 的動作，也就是 fig13 到 fig22 的動作 (以下稱為對稱性) ，在經過多次的實驗之後，我們猜測這樣的移動方法是最快的方法。

首先得知這樣的操作一定會成功(因為對稱性)，又過程中為了動某個環，必須將所有小於欲動的環移到其中一根柱子上，所以每一個動作都是必要的動作，得到結論：這樣的操作是最快的操作。之後我們將以這種操作方式進行討論。

接下來為了計算移動所有環所需要的次數，先證明下列引理：

引理一：

定義 b_n 為 n 個環以新規則，換到另一根柱子的步數(只要求最大的環在最下面)。且定義 a_n 為 n 個環以新規則，換到另一根柱子的步數(環由上而下，從小而大)，可寫出下列遞迴關係式：

$$b_n = b_{n-1} + n \ ， \ b_1 = 1$$

$$a_n = 2b_{n-1} + 1$$

證明：

(1). $b_n = b_{n-1} + n \ ， \ b_1 = 1$

在欲將 n 個環按照新規則移到另一根柱子的過程中，必須先將所有小於 n 的環放到同一根柱子上，再移動第 n 環，接下來只要把所有小於 n 的環直接移到放第 n 環的那根柱子上即可，所以共需

$$b_n = b_{n-1} + (n-1) + 1 = b_{n-1} + n$$

步。

(2). $a_n = 2b_{n-1} + 1$

在移動過程中欲將 n 個環按照大小排列移到另一根柱子，必須先將所有小於 n 的環放到同一根柱子上，再移動第 n 環，接下來只要把所有小於 n 的環以新規則移到放第 n 環的那根柱子上(逆操作)，所以共需

$$a_n = b_{n-1} + 1 + b_{n-1} = 2b_{n-1} + 1$$

步。

QED

定理一：

n 個環移動到另一根柱子所需的次數

$$a_n = n^2 - n + 1 = 2C_2^n + 1$$

證明：

根據引理一，我們可以求出 a_n 的遞迴式

$$\begin{aligned} a_n &= 2b_{n-1} + 1 \\ &= 2[b_{n-2} + (n-1)] + 1 \\ &= a_{n-1} + 2n - 2 \end{aligned}$$

解遞迴得到 a_n 一般式

$$a_n = n^2 - n + 1 = 2C_2^n + 1$$

QED

2、 搬動環的順序所成的數列及演算法

定義：$L_m(n)$ 為 m 個環的移動過程中，由搬動環的順序所成的數列。

由第一節的操作方法，我們列出以下幾個例子：

$L_2(n)$ 1 [2] 1

$L_3(n)$ 1 [2] 1 [3] 1 [2] 1

$L_4(n)$ 1 [2] 1 [3] 1 2 [4] 2 1 [3] 1 [2] 1

$L_5(n)$ 1 [2] 1 [3] 1 2 [4] 2 1 3 [5] 3 1 2 [4] 2 1 [3] 1 [2] 1

$L_6(n)$ 1 [2] 1 [3] 1 2 [4] 2 1 3 [5] 3 1 2 4 [6] 4 2 1 3 [5] 3 1 2 [4] 2 1 [3] 1 [2] 1

觀察以上結果，可以發現 $L_m(n)$ 數列不斷地對新出現的數字作對稱，因此我們定義每一個數字 x 第一次出現時，就稱 x 為**對稱點**，如 $L_3(n)$ 中的 2,3 皆為對稱點。符號 P_x 表示到達對稱點 x 的步數，如 $L_3(n)$ 中的 $P_3 = 4$。

因此我們有了以下的定理：

定理二：

出現 x（對稱點）的步數為

$$P_x = \frac{x^2 - x + 2}{2} = C_2^x + 1$$

證明：

整個數列的結構由前面可知，每出現一個新的數字 x，則 x 前的 $x-1$ 項對 x 作對稱，所以在出現 x 前已經動過

$$1 + 2 + 3 + \cdots\cdots + (x-1) = \frac{x(x-1)}{2}$$

步了，再加上新的數字「x」，總共 $\frac{x(x-1)}{2} + 1$ 步，也就是

$$\frac{x^2 - x + 2}{2} = C_2^x + 1$$

步。

QED

根據定理二及第(一)-1 節中逆操作所產生對稱的觀念，可以設計出下列演算法：

Algorithm $L(m,n)$

1. $L(m,1) \leftarrow 1$
2. *if* $n > \frac{m^2-m+2}{2}$ *then* $n \leftarrow m^2-m+2-n$
3. *for* $i = 1$ *to* n {
4. $L(m, \frac{i^2-i+2}{2}) \leftarrow i$
5. *for* $j = i-1$ *to* 1 *step* -1
6. $L(m, \frac{i^2-i+2}{2}+j) \leftarrow L(m, \frac{i^2-i+2}{2}-j)$
7. }

Algorithm $L(m,n)$ 分析：

Algorithm $L(m,n)$	Steps	說明
1. $L(m,1) \leftarrow 1$	$O(1)$	此演算法是得出 $L_m(n)$ 的非遞迴演算法。 作法： 定義第 1 步為 1，且找出所有的對稱點，第 i 個對稱點前 $i-1$ 個數對第 i 個對稱點作對稱，以此得到整串數列。
2. *if* $n > \frac{m^2-m+2}{2}$ *then* $n \leftarrow m^2-m+2-n$	$O(1)$	
3. *for* $i = 1$ *to* n {	$O(n)$	
4. $L(m, \frac{i^2-i+2}{2}) \leftarrow i$	$O(1)$	
5. *for* $j = i-1$ *to* 1 *step* -1	$O(n^2)$	
6. $L(m, \frac{i^2-i+2}{2}+j) \leftarrow L(m, \frac{i^2-i+2}{2}-j)$	$O(1)$	
7. }		
Total	$O(n^2)$	

3、 第 *n* 步的動作

在這一節中，我們主要探討第 n 步動作所動的環(以下所討論的環皆為無限多，因為若環的個數為有限個，則在最大環之後的動作皆可經由一次對稱得知，故討論無限個環的狀況)，我們將會分兩個方面去討論：

(1) 演　算　法：主要利用對稱性，設計出演算法。

(2) 數學一般式：利用對稱性再推導出 $L_m(n)$ 的另外性質，接著再引出一個和 $L_m(n)$ 中對稱點 x 後 $x-1$ 項相同的函數，所以只要利用夾擊求出 n 之前最大的對稱點，即可求出第 n 次動作所動的環。

(1)演算法

Algorithm L - 2(n)

1. *for* $i = 1$ *to* n*{*
2. *if* $\frac{i^2 - i + 2}{2} = n$ *then {*
3. *Return i*
4. *exit for*
5. *else if* $\frac{i^2 - i + 2}{2} > n$ *then*
6. $symmetry \leftarrow i - 1$
7. *exit for*
8. *}*
9. $n \leftarrow (n - (n - \frac{symmetry^2 - symmetry + 2}{2}))$
10. *}*

演算法分析：

Statement	Steps	說明
1. *for* $i = 1$ *to* n*{*	$O(n)$	此演算法為求出第 n 步動作的遞迴演算法。 作法： 先找出 n 之前最大的對稱點。接下來將 n 依序對之前的對稱點作對稱，當停在某個對稱點上，則第 n 次動作就是動那個環。
2. *if* $\frac{i^2 - i + 2}{2} = n$ *then {*	$O(1)$	
3. *Return i*	$O(1)$	
4. *exit for*	$O(1)$	
5. *else if* $\frac{i^2 - i + 2}{2} > n$ *then*	$O(1)$	
6. $symmetry \leftarrow i - 1$	$O(1)$	
7. *exit for*	$O(0)$	
8. *}*		
9. $n \leftarrow (n - (n - \frac{symmetry^2 - symmetry + 2}{2}))$	$O(1)$	
10. *}*		
Total	$O(n^2)$	

接下來利用數學的方法慢慢導出 $L_m(n)$：我們回憶一下 $L_m(n)$ 為有 m 個環移動過程中，搬動環的順序所成的數列。經過一些觀察，我們的策略是：

首先找出在第 n 步之前最大的對稱點，再構造等於 $L_m(n)$ 中對稱點 x 之後的 $x-1$ 項的數列 $T(x,i)$，就可以得到第 n 步動作所動的環。

引理二：

給定n值，求出最近的對稱點為x環

$$x=\left\lfloor\frac{-1+\sqrt{1+8(n-1)}}{2}\right\rfloor+1$$

證明：

$$P_x\le n<P_{x+1}\qquad\text{利用夾擊找出最近的對稱點}x$$

$$\frac{x^2-x+2}{2}\le n<\frac{x^2+x+2}{2}$$

$$\begin{cases}\dfrac{1-\sqrt{1+8(n-1)}}{2}\le x\le\dfrac{1+\sqrt{1+8(n-1)}}{2}\\ x>\dfrac{-1+\sqrt{1+8(n-1)}}{2}\bigcup x<\dfrac{-1-\sqrt{1+8(n-1)}}{2}\end{cases}$$

$$\frac{-1+\sqrt{1+8(n-1)}}{2}<x\le\frac{1+\sqrt{1+8(n-1)}}{2}$$

$$x=\left\lfloor\frac{-1+\sqrt{1+8(n-1)}}{2}\right\rfloor+1$$

QED

接下來構造一個新的數列$T(x,i)$，此數列共有$x-1$項，前$\left\lfloor\frac{x}{2}\right\rfloor$項為$x$不斷的減2，後$\left\lfloor\frac{x}{2}\right\rfloor$項則依據$x$的奇偶性，分別從0($x\equiv1(\bmod 2)$)或-1($x\equiv0(\bmod 2)$)不斷加2。

引理三：

數列$T(x,i)$如以下形式：

$x-2$、$x-4$、$x-6$、……、1、2、4、6、……、$x-1$，　當x為奇數時

$x-2$、$x-4$、$x-6$、……、2、1、3、5、……、$x-1$，　當x為偶數時

$$T(x,i)=\left|x-2i\right|+\left\lfloor\frac{i}{\left\lfloor\frac{x+1}{2}\right\rfloor}\right\rfloor$$

證明：

因為$T(x,i)$前$\left\lfloor\frac{x}{2}\right\rfloor$項為不斷的減2，後$\left\lfloor\frac{x}{2}\right\rfloor$項則為不斷加2，所以前$\left\lfloor\frac{x}{2}\right\rfloor$項的奇偶性恰好和後$\left\lfloor\frac{x}{2}\right\rfloor$項是相反的，也就是說，只需要對於$x$不斷的減2，第幾項就減幾個2，當出來的值為負值時就取絕對值(因為$+k\equiv-k(\bmod 2)$)，但是我們知道當出來的值為負值時表示已經到了後$\left\lfloor\frac{x}{2}\right\rfloor$項，所以奇偶性需要改變，而改變的方法就是加1(利用高斯的特性)。

QED

定理三：

$T(x,i)$ 符合原先 $L_m(n)$ 中對稱點 x 後的 x-1項

證明：

由 Knuth 河內塔的移動過程著手，說明在 $L_m(n)$ 中具有數列 $T(x,i)$ 的特性。

數列 $T(x,i)$ 特性：

(1) $T(x,i)>2$ ， $|T(x,i)-T(x,i-1)|=2$

在移動過程中，為了移動下一個第 n 環，必須將所有小於 n 的環先移到 n-1 環上，再將第 n 環移到空的柱子上(fig24 的動作)，接著為了動第 n+1 環，必須將小於 n 的環全部移到 n 環的上面，這個動作的最後一步，是動第 n-1 環，接下來就要動第 n+1 環了，由此可知，動完第 k 環，接下來不是動第 k+2 環，就是動第 k-2 環，(因為對稱性)，可知任意兩步之間，環的編號差 2。

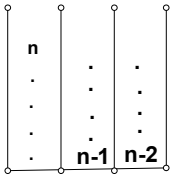


fig23

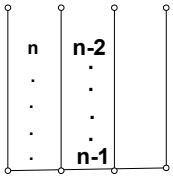


fig24

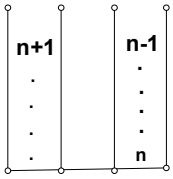


fig25

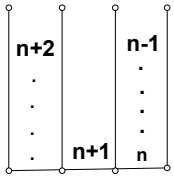


fig26

(2) $T(x,i)=1\rightarrow T(x,i+1)=2$ ，當 x 為奇數時

$T(x,i)=2\rightarrow T(x,i+1)=1$ ，當 x 為偶數時

這和一開始的移動有關，第一步動第 1 環，第 2 步動第 2 環，這是移動過程中唯一不是差 2 的部分，可由(1)知道這只是數列中的一個特例。

經由以上兩點我們可以知道數列 $L_m(n)$ 對稱點 x 後的 x-1 項列為 $T(x,i)$ 。

定理四：

x 為最靠近第 n 步之對稱點， $L_m(n)$ 即為所求：

$$L_m(n)=|x-2i|+\left\lfloor\frac{i}{\left\lceil\frac{x+1}{2}\right\rceil}\right\rfloor\ ,\ x=\left\lceil\frac{-1+\sqrt{1+8(n-1)}}{2}\right\rceil+1\ ,\ i=n-P_x\ ,\ P_x=C_2^x+1$$

證明：

給定 n 值之後，利用引理二找出最近的對稱點 x，再利用定理二求出 x 的步數 P_x，再算出 n 到 P_x 的差距 i，最後利用定理三及引理三即可求出

$$L_m(n)=|x-2i|+\left\lfloor\frac{i}{\left\lceil\frac{x+1}{2}\right\rceil}\right\rfloor$$

QED

因為數學的一般式已經做出來了，如果用這個一般式寫成演算法的話，時間複雜度為何呢？

Algorithm L-1(n)

1. $x \leftarrow Int([(-1)+\frac{Sqrt(1+8\times(n-1))}{2}])+1$
2. $i \leftarrow n-\frac{x\times(x-1)}{2}-1$
3. $L\text{-}1(n) \leftarrow Abs(x-2\times i)+Int(i/Int(\frac{x+1}{2}))$

演算法分析：

Statement	Steps	說明
1. $x \leftarrow Int([(-1)+\frac{Sqrt(1+8\times(n-1))}{2}])+1$	$O(1)$	經由數學方法的導證之後，將同樣一個動作從 $O(n^2)$ 簡化為 $O(1)$，在此可見數學威力之一斑。
2. $i \leftarrow n-\frac{x\times(x-1)}{2}-1$	$O(1)$	
3. $L\text{-}1(n) \leftarrow Abs(x-2\times i)+Int(i/Int(\frac{x+1}{2}))$	$O(1)$	
Total	$O(1)$	

4、 第 m 環何時被動到

用定理二可以先找出第一次動第 m 環是第幾步，接下來因為此數列不斷的以下一個新出現的環作對稱點，所以再找出第 $m+1$ 環第一次出現的步數，以此作對稱，求得下一次動 m 環的步數。以此做下去，即可得到第 m 個環何時被動到，依此可寫出下列演算法：

Algorithm $g(m,n)$

1. $g(m,1) \leftarrow (m^2-m+2)/2$
2. *for* $i=m+1$ *to* n*{*
3. $symmetry \leftarrow (i^2+i+2)/2$
4. $g(m,i-m+1) \leftarrow symmetry+symmetry-g(m,i-m)$
5. *}*

演算法分析：

Statement	Steps	說明
1. $g(m,1) \leftarrow (m^2-m+2)/2$	$O(1)$	作法： 已知第 m 環第一次動的步數，再對 m 之後的對稱點作對稱即可。
2. *for* $i=m+1$ *to* n*{*	$O(n)$	
3. $symmetry \leftarrow (i^2+i+2)/2$	$O(1)$	
4. $g(m,i-m+1) \leftarrow symmetry+symmetry-g(m,i-m)$	$O(1)$	
5. *}*	$O(0)$	
Total	$O(n)$	

(二)　Knuth 河內塔與新的正整數分割

藉由以上的結構分析，我們發現，若將環每次移動的過程利用前面所設計的演算法記錄下來，加以整理之後，對於正整數有一個良好的同餘循環性質，這在數論上有著其重要的意義。

1、 新的正整數分割

對於第 m 個環在何時移動，有另外的想法，首先舉一個 10 個環的例子：

n \ m	1	2	3	4	5	6	7	8	9	10
1	***1***	***2***	***4***	***7***	***11***	***16***	***22***	***29***	***37***	***46***
2	***3***	***6***	***10***	***15***	***21***	***28***	***36***	***45***	***55***	
3	***5***	***8***	***12***	***17***	***23***	***30***	***38***	***47***		
4	***9***	***14***	***20***	***27***	***35***	***44***	***54***	63		
5	***13***	***18***	***24***	***31***	***39***	***48***	56			
6	***19***	***26***	***34***	***43***	***53***	62	70			
7	***25***	***32***	***40***	***49***	57	64				
8	***33***	***42***	***52***	61	49	76				
9	***41***	***50***	58	65	71					
10	***51***	60	68	75	81					
11	59	66	72	77						
12	67	74	80	85						
13	73	78	82							
14	79	84	88							
15	83	86								
16	87	90								
17	89									
18	91									

表一(第 m 環第 n 次移動到的步數)

注意：這一節中我們所討論的表，只有表一斜體加粗字的部分。也就是討論無限多環的情形。

定義符號 $g(m,n)$ 為第 m 環第 n 次被移動到的步數，接下來我們試著求出 $g(m,n)$ 一般式：

首先將表一中相鄰兩欄之差列成表二和表三：

1	2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10	
3	4	5	6	7	8	**9**		
5	6	7	8	9	***10***	9		
5	6	7	8	9	8			
7	8	9	***10***	7	8			
7	8	***9***	8	7				
9	8	9	8	5				
9	8	7	6					
9	6	7	6					
7	6	5						
7	4	5						
5	4							
5	4							
3								
3								

表二

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

表三（表二組成）

先看表二的部分，除了第一列是 123456…外，接下來第二列就是第一列加 2，第三列等於第二列，第四列又等於第三列+2，由此猜測以下關係式：

第 $2s$ 列 = 第 $2s$-1 列+2

第 $2s+1$ 列 = 第 $2s$ 列

為了證明以上這個關係式，我們必須先證明一個引理，構造

$$T(x,i)=m \text{ 的反函數} \rightarrow T^{-1}(x,m)=i$$

引理四：

$T(x,i)=m$ 的反函數$\rightarrow T^{-1}(x,m)=i$

$$T^{-1}(x,m)=\begin{cases}[\frac{x}{2}]-[\frac{m}{2}], x\equiv m \pmod 2\\ [\frac{x}{2}]+[\frac{m}{2}], x\equiv m+1 \pmod 2\end{cases}$$

證明：

根據引理三，得知若 $x \equiv m(\bmod 2)$，則 m 出現在前 $\left[\frac{x}{2}\right]$ 內，又因為 $T(x,i)$ 前 $\left[\frac{x}{2}\right]$ 項不斷的減 2，所以得知

$$T^{-1}(x,m) = \left[\frac{x}{2}\right] - \left[\frac{m}{2}\right], x \equiv m(\bmod 2)$$

根據引理三，得知若 $x \equiv m+1(\bmod 2)$，則 m 會出現在後 $\left[\frac{x}{2}\right]$ 項，又因為 $T(x,i)$ 後 $\left[\frac{x}{2}\right]$ 項不斷的減 2，所以得知

$$T^{-1}(x,m) = \left[\frac{x}{2}\right] + \left[\frac{m}{2}\right], x \equiv m+1(\bmod 2)$$

QED

定理五：

定義 $t_{p\to p+1}$ 為第 t 環動第 $p+1$ 次和動第 p 次的步數差，k 為對稱點

$$k = t + p - 1$$

$$(t+1)_{p\to p+1} - t_{p\to p+1} = \begin{cases} 2, k \equiv t & (\bmod 2) \\ 0, k \equiv t+1 & (\bmod 2) \end{cases}$$

證明：

我們將任意對稱點 x 後 $x-1$ 項當作一組，這種分組方式可以讓我們知道任意環當時的移動次數(經由對稱性)。在每次移動第 t 環時，若第 t 環為第 p 次移動，則第 t 環之前最近的對稱點為第 $(t+p-1)$ 環，也就是 $k = t+p-1$。

並且我們知道關係式

第 $2s$ 列 = 第 $2s$-1 列+2

第 $2s+1$ 列 = 第 $2s$ 列

和定理五是一樣的意思。

因為關係式可表示如下：

$$\begin{aligned} &[g(t+1,p+1) - g(t,p+1)] - [g(t+1,p) - g(t,p)] \\ &= [g(t+1,p+1) - g(t+1,p)] - [g(t,p+1) - g(t,p)] \\ &= (t+1)_{p\to p+1} - t_{p\to p+1} \end{aligned}$$

首先將 $L_m(n)$ 數列中一段寫於下圖 fig27，k 表示對稱點 k，**t** 表示第 t 環，**p** 表示第 t 環動了第 p 次。

$L_m(n)$ 數列：……k……**t**…$t+1$…　k+1…$t+1$…**t**……　k+2……**t**…$t+1$…

使用次數：　**p**　$p-1$　　p　**p+1**　　**p+2**　$p+1$

fig27

分兩種情形討論：

(1)$k \equiv t(\bmod 2)$ 時

$L_m(n)$ 中取一段出來，如 fig27，分別計算出 $t_{p\to p+1}$ 和 $(t+1)_{p\to p+1}$，並計算兩者之差，計算過程中利用引理四可以算出步數之差。

$$t_{p\to p+1}=k+T^{-1}(k+1,t)+1-T^{-1}(k,t)-1$$
$$(t+1)_{p\to p+1}=(k+1)+T^{-1}(k+2,t+1)+1-T^{-1}(k+1,t+1)-1$$
$$(t+1)_{p\to p+1}-t_{p\to p+1}=(k+1+\left[\frac{k+2}{2}\right]+\left[\frac{t+1}{2}\right]+1-\left[\frac{k+1}{2}\right]+\left[\frac{t+1}{2}\right]-1)$$
$$-(k+\left[\frac{k+1}{2}\right]+\left[\frac{t}{2}\right]+1-\left[\frac{k}{2}\right]+\left[\frac{t}{2}\right]-1)$$
$$(t+1)_{p\to p+1}-t_{p\to p+1}=\left[\frac{k+2}{2}\right]+2\left[\frac{t+1}{2}\right]-2\left[\frac{k+1}{2}\right]-2\left[\frac{t}{2}\right]+\left[\frac{k}{2}\right]+1$$

$t\equiv 0(\mathrm{mod}\,2)$

$$(t+1)_{p\to p+1}-t_{p\to p+1}=\frac{k+2}{2}+t-k-t+\frac{k}{2}+1$$
$$=2$$

$t\equiv 1(\mathrm{mod}\,2)$

$$(t+1)_{p\to p+1}-t_{p\to p+1}=\frac{k+1}{2}+(t+1)-(k+1)-(t-1)+\frac{k-1}{2}+1$$
$$=2$$

$L(n)$ 數列：……k……**t+1**…t…　k+1…t…**t+1**……　k+2……**t+1**…t…

使用次數：　**p-1**　p　$p+1$　**p**　**p+1**　$p+2$

fig28

$(2)k\equiv t+1(\mathrm{mod}\,2)$ 時

$L_m(n)$ 中取一段出來，如 fig28，分別計算出 $t_{p\to p+1}$ 和 $(t+1)_{p\to p+1}$，並計算兩者之差，計算過程中利用引理二可以算出步數之差。

$$t_{p\to p+1}=k+T^{-1}(k+1,t)+1-T^{-1}(k,t)-1$$
$$(t+1)_{p\to p+1}=(k+1)+T^{-1}(k+2,t+1)+1-T^{-1}(k+1,t+1)-1$$
$$(t+1)_{p\to p+1}-t_{p\to p+1}=(k+1+\left[\frac{k+2}{2}\right]+\left[\frac{t+1}{2}\right]+1-\left[\frac{k+1}{2}\right]+\left[\frac{t+1}{2}\right]-1)$$
$$-(k+\left[\frac{k+1}{2}\right]+\left[\frac{t+1}{2}\right]+1-\left[\frac{k}{2}\right]+\left[\frac{t}{2}\right]-1)$$
$$=2\left[\frac{k+1}{2}\right]+2\left[\frac{t+1}{2}\right]-\left[\frac{k}{2}\right]-\left[\frac{k+2}{2}\right]-2\left[\frac{t}{2}\right]-1$$

$t\equiv 0(\mathrm{mod}\,2)$

$$(t+1)_{p\to p+1}-t_{p\to p+1}=k+t+2-\frac{k}{2}-\frac{k+2}{2}-(t-1)-1$$
$$=0$$

$t\equiv 1(\mathrm{mod}\,2)$

$$(t+1)_{p\to p+1}-t_{p\to p+1}=k+(t+2)-\frac{k}{2}-\frac{k+2}{2}-(t-1)-1$$
$$=0$$

經由上述繁瑣的計算及討論後得知

$$(t+1)_{p\to p+1}-t_p=\begin{cases}2, k\equiv t \pmod 2\\ 0, k\equiv t+1 \pmod 2\end{cases}$$

QED

定理六：

第 m 環第 n 次被移動的步數公式

1. $g(m,n)=\dfrac{(n+m-1)(n+m-2)}{2}+1+\left[\dfrac{n+m-1}{2}\right]+(-1)^n\left[\dfrac{m}{2}\right]$

2. $g(m,n)=\dfrac{n(n-1)}{2}+[\dfrac{n}{2}]+1+\dfrac{(2n+m-2)(m-1)}{2}+\dfrac{(m-1)[1+(-1)^n]}{2}$

證明：

我們先求出 $g(1,n)$ 的公式：

首先觀察第 1 環每次動到的步數差：2,2,4,4,6,6,8,8,10……

並由此推測移動第 1 環的步數公式：

$$g(1,n)=\frac{n(n-1)}{2}+[\frac{n}{2}]+1$$

$g(1,n)$ 的證明：

在每次動到第 1 環時，若第 1 環之前最近的對稱點為 n ，則第一環為第 n 次移動。對稱點 n 的步數為

$$\frac{n(n-1)}{2}+1$$

且第 1 環離對稱點 n 的步數為

$$T^{-1}(n,1)=\left[\frac{n}{2}\right]+(-1)^n\left[\frac{1}{2}\right]=\left[\frac{n}{2}\right]$$

兩者相加即為第 1 環第 n 次移動的步數。

QED

接下來，我們利用兩種作法求出 $g(m,n)$ ，兩種做法的表示方式也有所差異：

定理六-1：

第 m 環第 n 次被移動的步數公式

$$g(m,n)=\frac{(n+m-1)(n+m-2)}{2}+1+\left[\frac{n+m-1}{2}\right]+(-1)^n\left[\frac{m}{2}\right]$$

證明：

因為在每次移動第 m 環時，若第 m 環之前最近的對稱點為第 $(n+m-1)$ 環，則第 m 環為第 n 次移動。而對稱點步數為

$$\frac{(n+m-1)(n+m-2)}{2}+1$$

且第 m 環離最近對稱點 $(n+m-1)$ 的步數為

$$T^{-1}(n+m-1,m)$$

關於 $T^{-1}(n+m-1,m)$，我們可將本來分奇偶性討論的地方合併為一項：

$$T^{-1}(n+m-1,m)=\left[\frac{n+m-1}{2}\right]+(-1)^n\left[\frac{m}{2}\right]$$

最後將兩者相加得到

$$g(m,n)=\frac{(n+m-1)(n+m-2)}{2}+1+\left[\frac{n+m-1}{2}\right]+(-1)^n\left[\frac{m}{2}\right]$$

QED

定理六-2：

第 m 環第 n 次被移動的步數公式

$$g(m,n)=\frac{n(n-1)}{2}+[\frac{n}{2}]+1+\frac{(2n+m-2)(m-1)}{2}+\frac{(m-1)[1+(-1)^n]}{2}$$

證明：

已知 $g(1,n)$ 的公式和定理五、定理二之後，我們可以利用遞推的方式將 $g(m,n)$ 一般式寫出來，對於以上的情形有下列的遞迴關係式：

$$g(m,n)=g(m-1,n)+n+(m-2)+\frac{1+(-1)^n}{2}$$

遞迴式證明：

$$g(1,n)=\frac{n(n-1)}{2}+[\frac{n}{2}]+1$$

$$g(2,n)=g(1,n)+n+\left(\frac{1^n+(-1)^n}{2}\right)$$

……

$$g(m,n)=g(m-1,n)+(m+n-2)+\left(\frac{1^n+(-1)^n}{2}\right)$$

首先我們由定理二，求出第 m 環與第 $m-1$ 環第一次移動的步數差為 $m-1$，可以得到

$$g(m,1)=g(m-1,1)+(m-1)$$

的關係，且再經由定理五，並將分奇偶討論的地方合併後得到

$$g(m,n)=g(m-1,n)+(m+n-2)+\left(\frac{1^n+(-1)^n}{2}\right)$$

最後解遞迴式得到 $g(m,n)$ 一般式：

$$g(m,n)=\frac{n(n-1)}{2}+[\frac{n}{2}]+1+\frac{(2n+m-2)(m-1)}{2}+\frac{(m-1)[1+(-1)^n]}{2}$$

QED

我們最後再來探討表一的結構：

n \ m	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	***1***	***2***	***4***	***7***	***11***	***16***	***22***	***29***	***37***	***46***
2	***3***	***6***	***10***	***15***	***21***	***28***	***36***	***45***	***55***	
3	***5***	***8***	***12***	***17***	***23***	***30***	***38***	***47***		
4	***9***	***14***	***20***	***27***	***35***	***44***	***54***	63		
5	***13***	***18***	***24***	***31***	***39***	***48***	56			
6	***19***	***26***	***34***	***43***	***53***	62	70			
7	***25***	***32***	***40***	***49***	57	64				
8	***33***	***42***	***52***	61	49	76				
9	***41***	***50***	58	65	71					
10	***51***	60	68	75	81					
11	59	66	72	77						
12	67	74	80	85						
13	73	78	82							
14	79	84	88							
15	83	86								
16	87	90								
17	89									
18	91									

表一

請注意斜體加粗字，並請斜斜的看，他是由右上至左下，在左下至右上的排列，每次跳一格，並完全佈滿了正整數。

其實當 m 極大的時候，由 1 到 m 的環被動的步數，必為正整數，而且他將正整數劃分成了 m 類；同理，將此表的斜體加粗字做另一種分類，第 i 環的動作稱做數列 $<m=i>$，第 j 步的動作稱做數列 $<n=j>$。(以上所有定義的數列僅包括斜粗體字，討論 m 無限多的時候)在下一個段落中我們將討論它的特殊性質。

2、 模的性質和證明

接下來我們再看一個例子，當環無限多，我們將過程作成像表一的樣子，再取出斜線加深部分做成表四，這也就是我們定義的正整數分割，將整張表模 5 後再觀察：

1	2	4	7	11	16	22	29	37	46
3	6	10	15	21	28	36	45	55	66
5	8	12	17	23	30	38	47	57	68
9	14	20	27	35	44	54	65	77	90
13	18	24	31	39	48	58	69	81	94
19	26	34	43	53	64	76	89	103	118
25	32	40	49	59	70	82	95	109	124
33	42	52	63	75	88	102	117	133	150
41	50	60	71	83	96	110	125	141	158
51	62	74	87	101	116	132	149	167	186
61	72	84	97	111	126	142	159	177	196
73	86	100	115	131	148	166	185	205	226
85	98	112	127	143	160	178	197	217	238
99	114	130	147	165	184	204	225	247	270
113	128	144	161	179	198	218	239	261	284
129	146	164	183	203	224	246	269	293	318
145	162	180	199	219	240	262	285	309	334
163	182	202	223	245	268	292	317	343	370
181	200	220	241	263	286	310	335	361	388
201	222	244	267	291	316	342	369	397	426

表四

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

表五—左表 mod 5 之後

在以上的表格中，我們發現了一些規律：直行為 10 項一循環，橫列為 5 項一循環。寫出程式經過多次的實驗(模不同的值)，發現這並不是一個巧合，而且這是必然的結果，因此我們有了以下的定理：

定理七：

1. 將數列 $< n = j >$ 中每一項 mod p（p 任意大於 2 之整數），則其結果每 p 項循環。
2. 將數列 $< m = i >$ 中每一項 mod k（k 任意大於 2 之整數），則其結果每 $2k$ 項循環。

為了證明模之後會循環，我們需要一個引理：

引理五：

對於 $f(x)=ax^2+bx+c$ ， a,b,c 為整數時，

$$f(x)\equiv f(x+p)\ (\text{mod}\ p)$$

證明：

$$\begin{aligned} 0 &\equiv 2apx+ap^2+bp\ (\text{mod}\ p)\\ ax^2+bx+c &\equiv (ax^2+bx+c)\ +2apx+ap2+bp\ (\text{mod}\ p)\\ ax^2+bx+c &\equiv a(x+m)^2+b(x+m)+c\ (\text{mod}\ p)\\ f(x) &\equiv f(x+p)\ (\text{mod}\ p) \end{aligned}$$

QED

定理七-1：

將數列 $<n=j>$ 中每一項 mod p（p 任意大於 2 之整數），則其結果每 p 項循環。

證明：

固定 n ，分成 $2r$ 和 $2r-1$ 兩類，代入 $g(m,n)$ 公式中，

$n=2r$ -1 時：

$$g(m,n)=2r^2-2r+1+\frac{(4r-4+m)(m-1)}{2}=\frac{m^2+(4r-5)+4r^2-8r+6}{2}$$

$n=2r$ 時：

$$g(m,n)=2r^2+1+\frac{(4r+m)(m-1)}{2}=\frac{m^2+(4r-1)m+4r^2-4r+2}{2}$$

因為 $g(m,n)\in$ 又

$$m^2+(4r-5)+4r^2-8r+6$$

$$m^2+(4r-1)m+4r^2-4r+2$$

上面兩式都是整係數二次多項式，所以會產生每 p 個數一循環。

QED

定理七-2：

將數列 $<m=i>$ 中每一項 mod k（k 任意大於 2 之整數），則其結果每 $2k$ 項循環。

證明：

同定理八-2 的證明方式，固定 m 討論，但是因為結果會產生分母為 2 的高斯項，所以模之後會 $2k$ 循環。

QED

(三)　Knuth 河內塔與費氏數列

就像傳統河內塔具有著其深厚的數學內涵，Knuth 河內塔有著其數學本質，我們發現 Knuth 河內塔完全等價於**費氏數列**的一種特殊排列，也就是費波那契真分數的排序，以下是我們的觀察過程與證明其等價的過程。

經過了仔細的觀察，我們發現上述的正整數分割和分子分母皆為費波那契數的真分數排序有著驚人的連結。

讓我來看一個清楚的例子：

令 $n=8$，費氏數列的前 8 項分別為：

$$1,1,2,3,5,8,13,21$$

把分子分母都是以上這 8 個數的真分數由小排到大，我們得到以下的數列：

$$\frac{1}{21}<\frac{1}{13}<\frac{2}{21}<\frac{1}{8}<\frac{3}{21}<\frac{2}{13}<\frac{1}{5}<\frac{3}{13}<\frac{5}{21}<\frac{2}{8}<\frac{1}{3}<\frac{3}{8}<\frac{8}{21}<\frac{5}{13}<\frac{2}{5}<\frac{1}{2}<\frac{3}{5}<\frac{8}{13}<\frac{13}{21}<\frac{5}{8}<\frac{2}{3}$$
$$1\quad 2\quad 3\quad 4\quad 5\quad 6\quad 7\quad 8\quad 9\quad 10\quad 11\quad 12\quad 13\quad 14\quad 15\quad 16\quad 17\quad 18\quad 19\quad 20\quad 21$$

仔細觀察後，我們發現了以下驚人的結果！！！

以 F(8) 為分母的數，序號為 1、3、5、9、13、19，恰好是正整數分割中的 $g(1,n)$

以 F(7) 為分母的數，序號為 2、6、8、14、18，恰好是正整數分割中的 $g(2,n)$

……

而以另一種角度來看：

以 F(3) 為分子的數，序號為 3、6、10、15、21，恰好是正整數分割中的 $g(m,1)$

以 F(2) 為分子的數，序號為 1、2、4、7、11、16，恰好是正整數分割中的 $g(m,2)$

……

對於這些驚人的結果，我們猜測出以下的定理：

定理八：

給定 n 值（$n\geq 3$），並定義函數 $Q(i,j)=\dfrac{F(i)}{F(j)}$，且 $n\geq j>i\geq 2$。

對所有符合條件的 $Q(i,j)$ 由小到大排序，並寫出序號，則：

$Q(i,j)$（j 為定值）之序號為 $g(n-j+1,j)$ 數列

$Q(i,j)$（i 為定值）之序號為 $g(i,i-1)$ 數列

這一節將說明 Knuth 河內塔等價於分子分母皆為費波那契數的真分數排序。

1、 觀察

為了要證明以上的定理，我們希望兩者之間的本質是相同的即可。所以我們做了以下的觀察：

我們再來看一次以下的排列：

$$\frac{1}{21}<\frac{1}{13}<\frac{2}{21}<\frac{1}{8}<\frac{3}{21}<\frac{2}{13}<\frac{1}{5}<\frac{3}{13}<\frac{5}{21}<\frac{2}{8}<\frac{1}{3}<\frac{3}{8}<\frac{8}{21}<\frac{5}{13}<\frac{2}{5}<\frac{1}{2}<\frac{3}{5}<\frac{8}{13}<\frac{13}{21}<\frac{5}{8}<\frac{2}{3}$$
$$1\quad 2\quad 3\quad 4\quad 5\quad 6\quad 7\quad 8\quad 9\quad 10\quad 11\quad 12\quad 13\quad 14\quad 15\quad 16\quad 17\quad 18\quad 19\quad 20\quad 21$$

這一次我們僅觀察分母的順序，並且將之轉化成費波那契數的項數：

$$21,13,21,8,21,13,5,13,21,8,3,8,21,13,5,2,5,13,21,8,3$$

$$F(8),F(7),F(8),F(6),F(8),F(7),F(5),F(7),F(8),F(6),F(4),$$
$$F(6),F(8),F(5),F(7),F(3),F(7),F(5),F(8),F(6),F(4)$$

把上一行中的$F(i)$的i單獨寫出來得到

8,7,8,6,8,7,5,7,8,6,4,6,8,7,5,3,5,7,8,6,4,

將上一排數字根據第幾大，寫下幾(如 8 是第 1 大，則寫 1)得到

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

此串數列恰好為上一節$L_m(n)$中正整數分割的部分。

觀察以上的結果，我們猜測它是對的，但是要把 Knuth 河內塔和費氏數列做一個連接，我們遇到了困難，因為我們雖然猜測費氏數列真分數排序就等價於 Knuth 河內塔，但是我們連它的構造方法都是利用很原始的慢慢排序，所以想直接證明它和 Knuth 河內塔的關係難度極高。

我們一開始嘗試用代數的方法去證明此猜測，但是遇到極大的困難，之後我們嘗試以$L_m(n)$的新構造法及$Q(i,j)$排序的構造方式來證明這兩者是等價的。

2、 $L_m(n)$的新構造法及費氏數列排序

我們以另一種方式來構造$L_m(n)$，在這裡以$L_7(n)$為例：

(步驟 1) 先寫出最大的數字 7。
(步驟 2) 經由對稱性可知 6 必須對 7 對稱，所以我們可放入 6。
(步驟 3) 同樣地，5 對 6 對稱，放入 5。
(步驟 4) 放入 4 時對 5 對稱，但因對稱性可知 4 必須在 5 和 6 之間，所以放入 4。
(步驟 5) 在 4 和 5 之間的序列放入 3 且對 4 對稱。
(步驟 6) 在 3 和 4 之間的序列放入 2 且對 3 對稱。
(步驟 7) 在 2 和 3 之間的序列放入 1 且對 2 對稱。完成$L_7(n)$。

（1）7
（2） 676
（3） 5657565
（4） 45464575464454
（5） 34353464357534643534 3
（6） 2324235324642357532464235324232
（7） 1213124213531246421357531246421353124213121
fig27

在此我們利用$L_m(n)$之中的已知對稱性質來構造$L_m(n)$，先放入最大的數m，再利用基本的對稱性質由大到小依序將小於m的數填入，並也基於對稱性給出限制，如在放入k時，必須要在$k+1$、$k+2$之間的序列作對稱，這個構造法對定理的證明極有幫助。

接下來我們要構造分子分母皆為費波那契數的真分數排序，需藉由下面的引理：

引理六：

對於任意正整數 $a>b, c>d$ 且 $\frac{b}{a}<\frac{d}{c}$ ，則 $\frac{b}{a}<\frac{b+d}{a+c}<\frac{d}{c}$

證明：

因為 $\frac{b}{a}<\frac{d}{c}$ ，所以 $bc<ad$

$$ab+bc<ab+ad \rightarrow \frac{b}{a}<\frac{b+d}{a+c}$$

$$cd+bc<cd+ad \rightarrow \frac{b+d}{a+c}<\frac{d}{c}$$

$$\frac{b}{a}<\frac{b+d}{a+c}<\frac{d}{c}$$

QED

接下來我們利用引理六來構造 $Q(i,j)$ 排序：

(步驟 1) 先寫出 $\frac{0}{1}$ 及 $\frac{1}{1}$，且 $\frac{0}{1}<\frac{1}{1}$ 。

(步驟 2) 所有的 $Q(i,3)$，有 $Q(2,3)=\frac{1}{2}$ ，利用引理六：

因為

$$\frac{0}{1}<\frac{0+1}{1+1}<\frac{1}{1}$$

，所以

$$\frac{0}{1}<\frac{1}{2}<\frac{1}{1}$$

，也就得到 $n=3$ 時排序。

(步驟 3) 所有的 $Q(i,4)$，有 $Q(2,4)$、$Q(3,4)$，各為 $\frac{1}{3}$、$\frac{2}{3}$ ，利用引理六：

因為

$$\frac{0}{1}<\frac{0+1}{1+2}<\frac{1}{2}<\frac{1+1}{2+1}<\frac{1}{1}$$

，所以

$$\frac{0}{1}<\frac{1}{3}<\frac{1}{2}<\frac{2}{3}<\frac{1}{1}$$

得到 $n=4$ 時排序。

(步驟 4) 所有的 $Q(i,5)$，有 $Q(2,5)$、$Q(3,5)$、$Q(4,5)$, 各為 $\frac{1}{5}$、$\frac{2}{5}$、$\frac{3}{5}$ ，利用引理四：

因為

$$\frac{0}{2}<\frac{0+1}{2+3}<\frac{1}{3}<\frac{1+1}{3+2}<\frac{1}{2}<\frac{1+2}{2+3}<\frac{2}{3}<\frac{1}{1}$$

所以

$$\frac{0}{2}<\frac{1}{5}<\frac{1}{3}<\frac{2}{5}<\frac{1}{2}<\frac{3}{5}<\frac{2}{3}<\frac{1}{1}$$

得到 $n=5$ 時排序。

（1） $\frac{0}{1}<\frac{1}{1}$

（2） $\frac{0}{1}<\frac{1}{2}<\frac{1}{1}$ $n=3$

（3） $\frac{0}{1}<\frac{1}{3}<\frac{1}{2}<\frac{2}{3}<\frac{1}{1}$ $n=4$

（4） $\frac{0}{2}<\frac{1}{5}<\frac{1}{3}<\frac{2}{5}<\frac{1}{2}<\frac{3}{5}<\frac{2}{3}<\frac{1}{1}$ $n=5$

（5） $\frac{0}{3}<\frac{1}{8}<\frac{1}{5}<\frac{2}{8}<\frac{1}{3}<\frac{3}{8}<\frac{2}{5}<\frac{1}{2}<\frac{3}{5}<\frac{5}{8}<\frac{2}{3}<\frac{1}{1}$ $n=6$

……

fig28

以此類推，我們可得到給定 n 時，所有 $Q(i,j)$ 的排序。

這個構造法的原理是不斷利用引理六，直到做到我們所要的 n 為止，即得出所有 $Q(i,j)$ 的排序。

此外需注意兩點：

(1) 步驟 1 寫出的 $\frac{0}{1}$、$\frac{1}{1}$ 並不符合 $Q(i,j)$ 的定義，所以在排序時並不將其計入，我們只是利用它們使得整個構造過程更完整呈現。

(2) 當我們新棑入 $Q(i,j)$ 時(j 為定值)，原先的 0 要改成 $\frac{0}{F(j-2)}$，目的也是為了要使整個構造過程更完整。

3、 定理證明

定理八：

給定 n 值（ $n\geq 3$ ），並定義函數 $Q(i,j)=\frac{F(i)}{F(j)}$，且 $n\geq j>i\geq 2$。

對所有符合條件的 $Q(i,j)$ 由小到大排序，並寫出序號，則：

$Q(i,j)$（ j 為定值）之序號為 $g(n-j+1,j)$ 數列

$Q(i,j)$（ i 為定值）之序號為 $g(i,i-1)$ 數列

證明：

我們比較上述兩種構造法的特性，並證明兩者等價：

（1） $L_m(n)$ 的新構造方式

利用對稱性，在放入某數 k 時，必須對 $k+1$ 對稱，且要放在 $k+1$、$k+2$ 序列之間。

（2）費波那契真分數排序的構造法(分母部分)

利用引理二，在放入 $Q(i,j)$ 時（ j 為定值），可以得知 $Q(i,j)$ 必須放在 $Q(i,j-1)$、$Q(i,j-2)$ 之間（因 $F(j)=F(j-1)+F(j-2)$ ）。

我們在此可以知道（1）、（2）兩者等價，因為這兩者的構造方式是一樣的，都是當放入一個新數字 t，必須放在差 1，差 2 的數字之間。

舉出 $L_5(n)$ 及分子分母皆為費波那契數的真分數排序 $n=6$ 的例子：

並對 $L_5(n)$ 中取對稱點 5(最大的對稱點)的前半段：

$$L_5(n) \quad 121312421353124213121$$

$$\rightarrow 1213124213$$

$$n=6 \quad \frac{1}{8}<\frac{1}{5}<\frac{2}{8}<\frac{1}{3}<\frac{3}{8}<\frac{2}{5}<\frac{1}{2}<\frac{3}{5}<\frac{5}{8}<\frac{2}{3}$$

容易看出若令 $Q(i,6)$ 為第 1 環，$Q(i,5)$ 為第 2 環，$Q(i,4)$ 為第 3 環，$Q(i,3)$ 為第 4 環，這時兩者是相同的。在此我們證明了 $Q(i,j)$ 之序號為 $g(n-j+1,j)$ 數列。

（3）費波那契真分數排序的構造法(分子部分)

在放入 $Q(i,j)$ 時（ j 為定值），可利用（2），找出排列 $Q(i,j)$ 的位置，再依序 將 $Q(1,\dot{j})$、$Q(2,j)$、$Q(3,j)$、…、Q(j-1,j) 填入位置中以完成排序，這種做法與引理二的構造法顯然是相同的。

經由（2）的證明我們知道 $Q(i,j)$（j 為定值）之序號為 $g(n-j+1,j)$ 數列，且依照（3）特性，知道 $Q(1,\dot{j})$ 為第 $(n-j+1)$ 環第 1 次被動到、$Q(2,\dot{j})$ 為第 $(n-j+1)$ 環第 2 次被動到…，$Q(j-1,\dot{j})$ 為第 $(n-j+1)$ 環第 $(j-1)$ 次被動到，故得知 $Q(i,j)$（ i 為定值）為第 j 環第 $(i-1)$ 次動到的步數數列，在此我們證明了 $Q(i,j)$（ i 為定值）之序號為 $g(i,i-1)$ 數列。

一樣舉出 $L_5(n)$ 及分子分母皆為費波那契數的真分數排序 $n=6$ 的例子：

並對 $L_5(n)$ 中取對稱點 5(最大的對稱點)的前半段：

$$L_5(n) \quad 121312421353124213121$$

$$\rightarrow 1213124213$$

$$n=6 \quad \begin{array}{ccccccccccc} \frac{1}{8} & < & \frac{1}{5} & < & \frac{2}{8} & < & \frac{1}{3} & < & \frac{3}{8} & < & \frac{2}{5} & < & \frac{1}{2} & < & \frac{3}{5} & < & \frac{5}{8} & < & \frac{2}{3} \\ & 1 & & 2 & & 1 & & 3 & & 1 & & 2 & & 4 & & 2 & & 1 & & 3 \end{array}$$

容易看出 $Q(i,j)$ 分子的數字代表該環的移動次數。在此證明了 $Q(i,j)$（ i 為定值）之序號為 $g(i,i-1)$ 數列。

QED

(四)　隨意亂排的 Knuth 河內塔復原演算法

在解決了 Knuth 河內塔問題的過程中，我們看到了一篇論文[4]，他給了一個可以快速將起始狀態為隨意亂排的傳統河內塔復原的演算法，他的主要策略是構造一個樹，此樹的每一個節點表示一種狀態，而只要從此樹找出最短路徑，則依照此最短路徑的移動方法即可達到最快復原的效果。

同樣的想法，我們想要移植到 Knuth 河內塔上，我們也需要找出一個圖與 Knuth 河內塔的每一種狀態對應，首先我們就要算出起始狀態總數，以方便找到某一個圖可以與之對應。

1.起始狀態計數

定理九：

n 個環的起始狀態總數為

$$(n-1)!\sum_{k=1}^{n-1}\frac{1}{k}$$

在這裡我們將條件嚴格限制：環可任意配置在其中兩根柱子上，但最大的環必須放在柱子底部。接下來我們將以排列組合計算，以期求出一般式並進而找出可能對應的圖形。

首先我們先決定兩根柱子中最大的環是哪兩個，有以下的情形：

$$(n,1),(n,2),(n,3),(n,4).....(n,n-1)$$

接下來個別討論：

$(n,1):(n-2)!0!$種

$(n,2):(n-2)!0!+C_1^1(n-3)!1!$種

$(n,3):(n-2)!0!+C_1^2(n-3)!1!+C_2^2(n-4)!2!$種

$(n,4):(n-2)!0!+C_1^3(n-3)!1!+C_2^3(n-4)!2!+C_3^3(n-4)!3!$種

$$\cdots\cdots$$

$(n,n-1):(n-2)!0!+C_1^{n-2}(n-3)!1!+C_2^{n-2}(n-4)!2!+.....+C_{n-2}^{n-2}0!(n-2)!$

對於各情形中繁雜的結果，我們利用以下的組合恆等式化簡：

$$C_n^n+C_n^{n+1}+C_n^{n+2}+.....+C_n^{n+k}=C_{n+1}^{n+k+1}$$

化簡後可得：

$$\begin{aligned}
&C_1^{n-1}(n-2)!0!+C_2^{n-1}(n-3)!1!+C_3^{n-1}(n-4)!2!+.....+C_{n-1}^{n-1}0!(n-2)!\\
&=\sum_{k=1}^{n-1}C_k^{n-1}(n-k-1)!(k-1)!\\
&=\sum_{k=1}^{n-1}\frac{(n-1)!}{k!(n-k-1)!}(n-k-1)!(k-1)!\\
&=\sum_{k=1}^{n-1}\frac{(n-1)!}{k}\\
&=(n-1)!\sum_{k=1}^{n-1}\frac{1}{k}
\end{aligned}$$

QED

此為總情形數，但遺憾的是，上式所得的式子並無法以一般式表示，也難以找出可對應的圖形，這只是初步的計量而已。

鑒於起始狀態總數的眾多，我們若想要設計出一個演算法可以針對每一種起始狀況進行最佳的恢復過程，則這個演算法所需要的時間複雜度大到難以想像，基於這樣的原因，我們設計了以下的演算法：

2. *Rug Paving Algorithm*

接下來，我們就著力於用演算法解決這個問題，我們設計出了「*Rug Paving Algorithm*」，這個演算法的主要策略是利用 *Divide Algorithm* 將最大的三個環分置於三根柱子的最下方，再利用 *Unrestricted Algorithm* 將其他的環(最大三個環以外的環)排序在第三大環所在的柱子上，最後利用 *Knuth*－1 *Algorithm* 將之完全復原即可。

A. *Divide Algorithm*

Divide Algorithm 主要的策略如下：

一、將所有的環擺到同一根柱子(並依照大小加以編號)

二、將所有的環依所在位置排序。

三、搜尋第二大的環做一個定位點，再搜尋第三大的環做一次定位點。

四、比較兩個定位點的位置，找出較上方定位點以上部份的第二大環做一個定位點。

五、找到新的定位點之後，在找出定位點以上最大的環作為定位點。

六、不斷的重複操作第五步的動作，直到沒有環的時候

七、將最上面的定位點移至第二根柱子。

八、將第一根柱子上最上方的定位點以上的環也都移到第二根柱子。

九、重複第七步、第八步的動作，直到最大的三個環分別在三根柱子的底端。

Divide Algorithm	*Analyze*		
1. *ring[0...2][0...n]*	$O(1)$	$O(1)$	定義變數
2. *k[0...2]*	$O(1)$		
3. *count←1*	$O(1)$		
4. *point[0...n]*	$O(1)$		
5. *max←0*	$O(1)$		
6. *for i = 0 to 2{*	$\theta(3)$	$O(1)$	定義柱子的編號
7. *if (ring[i][1]=1){*	$\Theta(1)$		
8. *k[0]←i*	$\Theta(1)$		
9. *else*	$\Theta(1)$		
10. *k[count]←i*	$\Theta(1)$		
11. *count←count+1*	$\Theta(1)$		
12. *}*			
13. *}*			
14. *for i = 1 to 2{*	$\Theta(2)$	$O(n)$	將所有的環擺到同一根柱子
15. *for j = len(ring[k[i]]) down to 0*	$O(n)$		
16. *MOVE(ring[k[i]][j],k[0])*	$\Theta(1)$		
17. *}*			
18. *count←0*	$\Theta(1)$	$\Theta(1)$	
19. *for j =2 to len(ring[k[0]]){*	$O(n)$	$O(n^2)$	尋找定位點
20. *i ←j*			
21. *while i < len(ring[k[0]]) do*	$O(n)$		
22. *max←Max(max,ring[k[0]][i])*	$\Theta(1)$		
23. *i ← i +1*			
24. *point[count]←max*	$\Theta(1)$		
25. *count←count+1*	$\Theta(1)$		
26. *max←0*	$\Theta(1)$		
27. *}*			
28. *i←1*	$\Theta(1)$	$\Theta(1)$	
29. *count←count-1*	$\Theta(1)$	$\Theta(1)$	
30. *while(ring[k[1]][0] × ring[k[2]][0]!=2) do{*	$O(n)$	$O(n^2)$	進行搬運
31. *for j = len(ring[k[0]]) down to point[count]*	$O(n)$		
32. *MOVE(ring[k[0][j]],i)*	$\Theta(1)$		
33. *}*			
34. *i←(i+1) mod 2*	$\Theta(1)$	$\Theta(1)$	
35. *count←count-1*	$\Theta(1)$	$\Theta(1)$	

B. *Unrestricted Algorithm*

Unrestricted Algorithm 主要策略如下：

一、將全部的環移到一根柱子上。

二、從 *n* 到一做第三步的動作。

三、找到所選取的環，並使此環上面所有的環移到第二根柱子上，再將此環移到要求的柱子上，並把第二根柱子上的環移回去第一根柱子。

Unrestricted Algorithm	*Analyze*		
1. *for i =1 to 2{*	$\Theta(2)$	$O(n)$	將全部的環移到一根柱子上
2. *for j = len(ring[k[i]]) down to 0*	$O(n)$		
3. *MOVE(ring[k[i]][j],k[0])*	$\Theta(1)$		
4. *}*			
5. *count←0*	$\Theta(1)$	$\Theta(1)$	
6. *for i = n down to 1{*	$O(n)$	$O(n^2)$	找到所選取的環，並使此環上面所有的環移到第二根柱子上，再將此環移到要求的柱子上，並把第二根柱子上的環移回去第一根柱子
7. *for j = len(ring[k[0]]) down to demand[count]*	$O(n)$		
8. *MOVE(ring[k[0]][i],k[2])*	$\Theta(1)$		
9. *MOVE(ring[k[0]][j],k[1])*	$\Theta(1)$	$\Theta(1)$	
10. *for j = len(ring[k[0]]) down to 0*	$O(n)$	$O(n)$	
11. *MOVE(ring[k[0]][j],k[1])*	$\Theta(1)$		
12. *count←count-1*	$\Theta(1)$	$\Theta(1)$	
13. *}*			

C. *Knuth*－1 *Algorithm*

Knuth－1 *Algorithm* 主要策略如下：

一、從 1 到 *n* -1 做第二步的動作。

二、將選取的環移到要求的柱子上後，再把比它小的環全部移到該根柱子。

三、將 *n* 移至第一根柱子上。

四、從 *n* -1 到 1 做第五步的動作。

五、選取環，並將比它小的環移到指定的柱子上，再將此環移到第一根柱子上。

Knuth－1 *Algorithm*	*Analyze*		
1. *for i = 1 to n-1*	$O(n)$	$O(n)$	將選取的環移到要求的柱子上
2. *MOVE(demand[count] ,demandcolumn)*	$\Theta(1)$		
3. *while ring[k[0]][i]<demand[count]*	$O(n)$	$O(n)$	把比它小的環全部移到該根柱子
4. *MOVE(ring[k[0]][i],demandcolumn)*	$\Theta(1)$		
5. *MOVE (n,1)*	$\Theta(1)$	$\Theta(1)$	移動
6. *for i = n-1 down to 1{*	$O(n)$	$O(n^2)$	選取環，並將比它小的環移到指定的柱子上，再將此環移到第一根柱子上
7. *while ring[k[0]][i]<demand[count]*	$O(n)$		
8. *MOVE(ring[k[0]][i],demandcolumn)*	$\Theta(1)$		
9. *MOVE(demand[count] ,demandcolumn)*	$\Theta(1)$		
10. *}*			

最後我們來探討 *Rug Paving Algorithm* 的時間複雜度，因為 *Divide Algorithm* 、 *Unrestricted Algorithm* 、 *Knuth*−1 *Algorithm* 的時間複雜度都不超過 $O(n^2)$ ，而且三個演算法是個別加成的，所以 *Rug Paving Algorithm* 的時間複雜度也為 $O(n^2)$ ，讓我們回憶一開頭所提及的初始狀態計數，他的數量為

$$(n-1)!\sum_{k=1}^{n-1}\frac{1}{k}$$

若我們要設計出一個演算法，它能夠針對每一種狀況進行最佳的復原操作，則這個演算法至少要能夠分辨出

$$(n-1)!\sum_{k=1}^{n-1}\frac{1}{k}$$

種初始狀態，從而得知：這個演算法的時間複雜度為 $O(n^n)$ ，如今我們設計出 *Rug Paving Algorithm*，這個演算法雖然無法針對每一種情況進行最佳的復原，但是他的時間複雜度大幅的下降成為 $O(n^2)$ ，而且對於復原操作亦不會有太多的無意義動作，對我們而言他算是一個好的演算法了！

3. *Divide Algorithm* 和 *Knuth Hanoi Graph*

A. *Knuth Hanoi Graph*

接下來我們嘗試觀察 *Divide Algorithm* 運作過程，我們發現了以下的規律：

我們給定一個符號 (i,j,k) 表示第一根柱子上有 i 個環，第二根柱子有 j 個環，第三根柱子有 k 個環，下圖為 *Knuth Hanoi Graph* ，簡稱 *KHG* ，是所有狀況移動過程所形成的圖。我們發現無論環和環之間是如何的擺置，我們只需要考慮每根柱子有幾個環即可，這樣的操作完全按照 *KHG* 來運行，每到一次交叉點，我們就有兩條路可以選擇，我們再依照實際狀況，選擇可繼續操作的路徑。如此操作下去，必然會在某個交叉點停住(最大的三個環都分別在三根柱子的底部)。如此，我們若能夠構造出 *KHG* ，則在 *Divide Algorithm* 中的排序就可以完全的省略了。

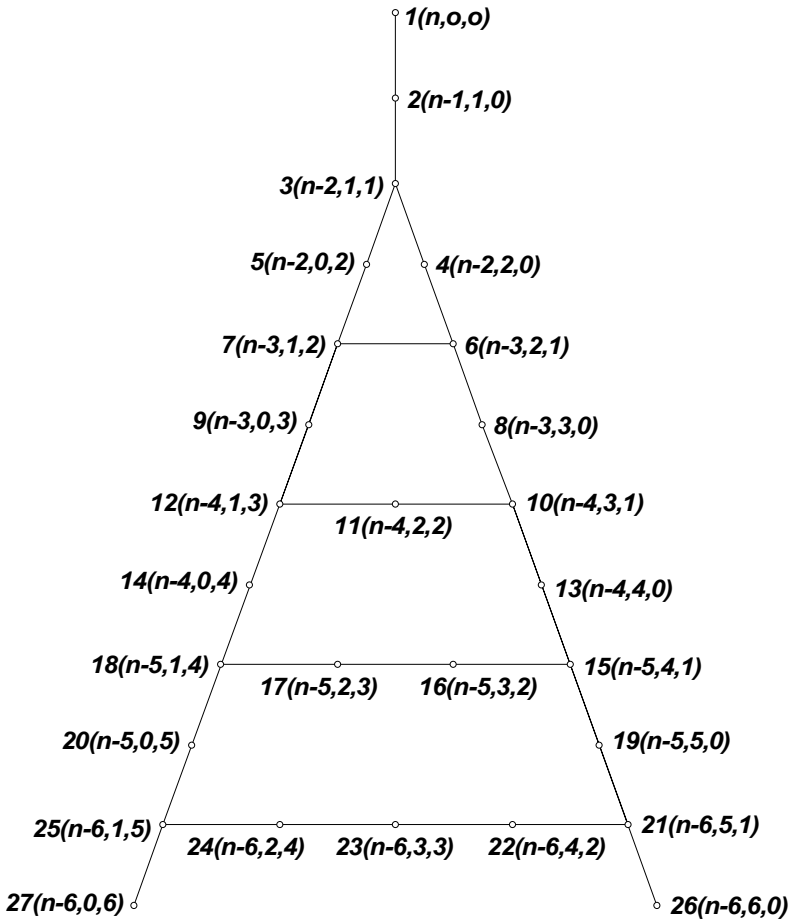


fig29 *KHG*

B. *KHG* 構造：

我們來考慮一下 *Knuth* 河內塔，因為第一個動作可以是將環移動到第二根柱子，或是將環移動到第三根柱子，而第一步的動作影響到了後面的移動方法，我們若是將兩種情況一併仿造 *KHG* 的做法作出一張圖，發現此圖就和 *KHG* 一樣了，表示 *Divide Algorithm* 和 *Knuth*－1 *Algorithm* 的操作過程中，環擺置狀態的改變是相同的，而在 *Rug Paving Algorithm* 中的最後一部份也是進行著 *Knuth*－1 *Algorithm* 的操作，因此無論如何我們都需要利用時間複雜度為 $O(n^2)$ 的 *Knuth*－1 *Algorithm* 來操作一次，我們不妨在運作的過程中先記錄下 *Knuth*－1 *Algorithm* 的每個狀態下環的配置情形，再利用這個記錄下來的資料所形成的 *KHG* 就可以在不用排序的情況下進行 *Divide Algorithm* 的部份了。

舉例如下：環數 $n=6$ 的時候的 *KHG*

1. 首先利用 *Knuth*－1 *Algorithm* 跑出資料(記錄兩種情況，分別是將第一環放置　　於第二根柱子或第三根柱子)，做出 *KHG* ：

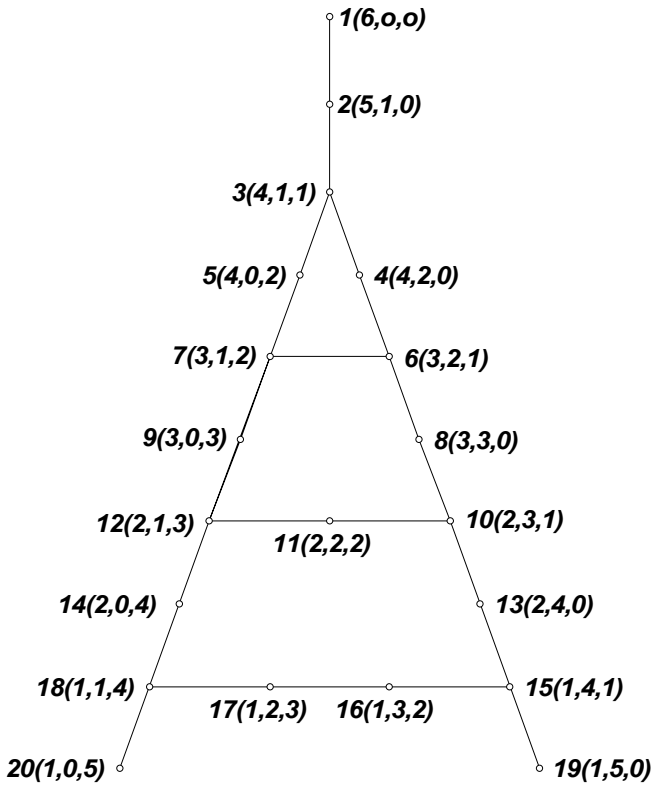


fig 30 *KHG*(6)

2. 接下來無論環如何的擺置就不斷的利用這張圖，由上往下，遇到 $degree(P)=3$ 的點 P 時，存在著唯一一條可以繼續移動的路徑。不斷的依照這個規則，直到停在某一個點，使得最大的三個環分別在三根柱子的最下方。

我們設計了以下的演算法：

KHG Algorithm	*Analyze*		
1. *struct KHG{*	*O*(1)	*O*(1)	定義陣列
2. *int a*	*O*(1)		
3. *int b*	*O*(1)		
4. *int c*	*O*(1)		
5. *}*			
6. $t \leftarrow 0$	*O*(1)		
7. *for i = 1 to n-1{*	*O*(*n*)	*O*(*n*)	進行 *Knuth* 河內塔的操作，並記錄之。
8. *t = t + 1*	*O*(1)		
9. *MOVE(demand[count] ,demandcolumn)*	*O*(1)		
10. *KHG[t], A←len(ring[k[0]])*	*O*(1)		
11. *KHG[t],B←len(ring[k[1]])*	*O*(1)		
12. *KHG[t],C←len(ring[k[2]])*	*O*(1)		
13. *}*			
14. *while ring[k[0]][i]<demand[count]{*	*O*(*n*)	*O*(*n*)	
15. *t = t + 1*	*O*(1)		
16. *MOVE(ring[k[0]][i],demandcolumn)*	*O*(1)		
17. *KHG[t], A←len(ring[k[0]])*	*O*(1)		
18. *KHG[t],B←len(ring[k[1]])*	*O*(1)		
19. *KHG[t],C←len(ring[k[2]])*	*O*(1)		
20. *}*			
21. *t = t + 1*	*O*(1)	*O*(1)	
22. *MOVE (n,1)*	*O*(1)		
23. *KHG[t], A←len(ring[k[0]])*	*O*(1)		
24. *KHG[t],B←len(ring[k[1]])*	*O*(1)		
25. *KHG[t],C←len(ring[k[2]])*	*O*(1)		
26. *for i = n-1 down to 1{*	*O*(*n*)	$O(n^2)$	
27. *while ring[k[0]][i]<demand[count]{*	*O*(*n*)		
28. *t = t + 1*	*O*(1)		
29. *MOVE(ring[k[0]][i],demandcolumn)*	*O*(1)		
30. *KHG[t], A←len(ring[k[0]])*	*O*(1)		
31. *KHG[t],B←len(ring[k[1]])*	*O*(1)		
32. *KHG[t],C←len(ring[k[2]])*	*O*(1)		
33. *t = t + 1*	*O*(1)		
34. *MOVE(demand[count] ,demandcolumn)*	*O*(1)		
35. *KHG[t], A←len(ring[k[0]])*	*O*(1)		
36. *KHG[t],B←len(ring[k[1]])*	*O*(1)		
37. *KHG[t],C←len(ring[k[2]])*	*O*(1)		
38. *}*			
39. *}*			

(五) 其他

如果我們將 Knuth 河內塔做一個推廣，我們定義 $Knuth_r$ 為在河內塔移動過程中，隨時保持每根柱子中最大的 r 個環依序排在最下方，則 $Knuth_r(n)$ 是否有一般式呢？

從以上的報告中我們已然的得知

$$Knuth_1(n) = n^2 - n + 1$$

，而且我們從傳統的河內塔也知道

$$Knuth_n(n) = 2^n - 1$$

如果 $Knuth_r(n)$ 的一般式存在著某種遞推關係，則這種關係竟將一個函數從多項式函數過渡到指數函數，這是一個怎麼樣的遞推關係呢？我們先藉由人工的方法找出 $Knuth_r(n)$ 的前幾個一般式，希望可以觀察這些一般式進而找出遞推關係。

(1) $Knuth_2(n)$ 的計算

我們首先觀察幾個例子：

當 $n=5$ 時，移動環的順序為：

1213121 4121312 5 2131214 1213121

當 $n=6$ 時，移動環的順序為：

1213121412321 51213124213 6 1242131215 1232141213121

在此我們可以發現 $Knuth_2$ 河內塔不但有逆對稱的情形，我們更可發現，移動過程是由兩段具有規律的數列所組成。尤其在第二段移動當中，因為三根柱子上最大的環已經擺在底部，所以剩下的操作過程和 $Knuth_1$ 河內塔的操作是完全相同的。據此我們可以寫出遞迴式算出 $Knuth_2(n)$：

定理十：

限制在同一根柱子中，最大的兩個環須依序排在最底部時，

n 個環移動到另一根柱子所需的次數

$$Knuth_2(n) = 3n^2 - 13n + 19$$

證明：

定義

c_n 為第一根柱子上還有兩個以上未移動的環時，n 個環移到另一根柱子的次數

b_n 為第一根柱子上只剩一個未移動的環時，n 個環移到另一根柱子的次數

a_n 為 $Knuth_2(n)$，為 n 個環按照規則限制，移動到另一根柱子所需的次數

則

$$a_n = 2(b_{n-1} + c_{n-2}) + 1$$
$$b_n = b_{n-1} + n + 1$$
$$c_n = c_{n-1} + (2n-2)$$

解遞迴，即可得到

$$a_n = 3n^2 - 13n + 19$$
$$Knuth_2(n) = 3n^2 - 13n + 19$$

QED

(2) $Knuth_r(n)$ 的推導：

我們重新定義符號：

$A^r{}_n$ 為 $Knuth_r(n)$

$B^r{}_n$ 為 $A^r{}_n$ 的第一段遞迴式

$C^r{}_n$ 為 $A^r{}_n$ 的第二段遞迴式

以此類推

以下是我們發現的關係：

$$\begin{aligned}
A^1{}_n &= 2B^1{}_{n-1}+1 \\
&= 2A^1{}_{n-1}+2(n-1) \\
B^1{}_n &= B^1{}_{n-1}+(n-1)+1 \\
A^2{}_n &= 2(B^2{}_{n-1}+C^2{}_{n-2})+1 \\
&= 2(A^1{}_{n-2}+B^1{}_{n-2})+3 \\
B^2{}_n &= B^2{}_{n-1}+n+1 \\
&= B^1{}_{n-1}+1 \\
C^2{}_n &= C^2{}_{n-1}+2n-2 \\
&= A^1{}_n
\end{aligned}$$

這樣的關係使我們以為可以簡單的利用一些遞迴式找出 $A^r{}_n$ 的一般式。所以我們接下來嘗試計算 $Knuth_3(n)$ 的一般式，發現 $Knuth_3(n)$ 應該是由三段遞迴式所構成的，他的第一、二段遞迴式我們已然找出，但是第三段遞迴式，再我們的嘗試之下仍無一般性的結果，這表示說，$Knuth_r(n)$ 的遞推關係，並非我們所想像的那麼簡單，可以預期的是，它存在著某個相變，使得 $Knuth_r(n)$ 的成長速度驚人。所以這個部分我們將之定為未來展望，希望可以找出 $Knuth_r(n)$ 的遞推關係，用來觀察 $Knuth_r(n)$ 在成長時所發生的相變。

三、結論與展望

在這篇報告中，我們對於 Knuth 河內塔做了核心的研究，對於他第幾次動作及哪一個環在何時移動都給了完整的探討，並依照 Knuth 河內塔的核心架構將正整數做了一個分割，及引出一些相關性質，並且證明 Knuth 河內塔等價於分子分母皆為費波那契數的真分數排序，以及完全解決了隨意亂排的 Knuth 河內塔復原問題，以下是幾個未來的展望：

1、 $Knuth_r(n)$ 的遞推關係與相變之觀察

2、 Knuth 河內塔問題與與圖論

3、 Knuth 河內塔問題與向量

四、研究心得

最後要感謝指導老師*-*，真正的帶了我們做了研究，過程中難免會 stuck，但是老師總會在後面推你一把，另外要感謝班上同學的熱心幫助。

五、參考資料及其他

[1] 莊益瑞，C++程式設計實務，台北，碁峰出版社
[2] 張鎮華，從演算法發現數學-河內塔面面觀，建中演講講稿，2002
[3] D. Knuth, All questions answered, *Notice A.M.S.* **49**(3), 318-324, 2002
[4] David G. Poole, The Towers and Triangle of Professor Claus(or, Pascal Knows Hanoi), **67**(5), 323-343, 1994
[5] Thomas H. Cormen, *Introduction To Algorithm*, II, London, The MIT Press, 123-195, 2001

Motivation

In a recent paper [3], Donald Knuth described the Tower of Hanoi as follows:

> ***"You have 3 pegs, and you have disks of different sizes. You're supposed to transfer the disks from one peg to another, and the disks have to be sorted on each peg so that the biggest is always on the bottom. You can move only one disk at a time."***

This apparent typo received the attention of John McCarthy to raise the query [5]: what if an order of disks on a peg is acceptable as long as the disk on the bottom is largest?

In this project, we solve the question completely, among other interesting findings.

Procedures

0. Notations

We summarize the rules in Tower of Hanoi a lá Knuth:

Rules

1) In initial state. You have 3 pegs A, B, C, and n disks of different sizes are placed in one peg in order of size.
2) You are supposed to transfer the disks from one peg to another, and you can move only one disk at a time.
3) On each peg, only the **biggest** disk is required placed on the bottom while the remaining disks may be placed in any order.

Throughout the discussion we let n denote the number of disks in the puzzle. The disks are labeled 1, 2, 3,⋯, n according to its ranking in size.

Let $L_n(k)$ denote the label of the disk to be moved at step k for a puzzle of n disks. We list several instances in Table 1.The red numbers represent the disk x is moved for the first time, they are symmetry point in $L_n(k)$.

Table 1(The sequence L_n)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
L_2	**1**	**2**	1																		
L_3	**1**	**2**	1	**3**	1	2															
L_4	**1**	**2**	1	**3**	1	2	**4**	2	1	3	1	2	1								
L_5	**1**	**2**	1	**3**	1	2	**4**	2	1	3	**5**	3	1	2	4	2	1	3	1	2	1

The method of moving disks is explained in the next section.

1. Intuitions

In Tower of Hanoi a lá Knuth, it is clear that in order to move disk *x* to another peg, we must:

1) move the set A of all disks above disk *x* to the unique available peg;
2) move disk *x*;
3) move all disks in *A* on top of disk *x*.

The disks in *A* are moved fastest follow the "LIFO" procedure (Last in, first out). It is also clear that once the biggest disk has been moved, the remaining steps are to be taken symmetrically.

If $n = 5$, then:

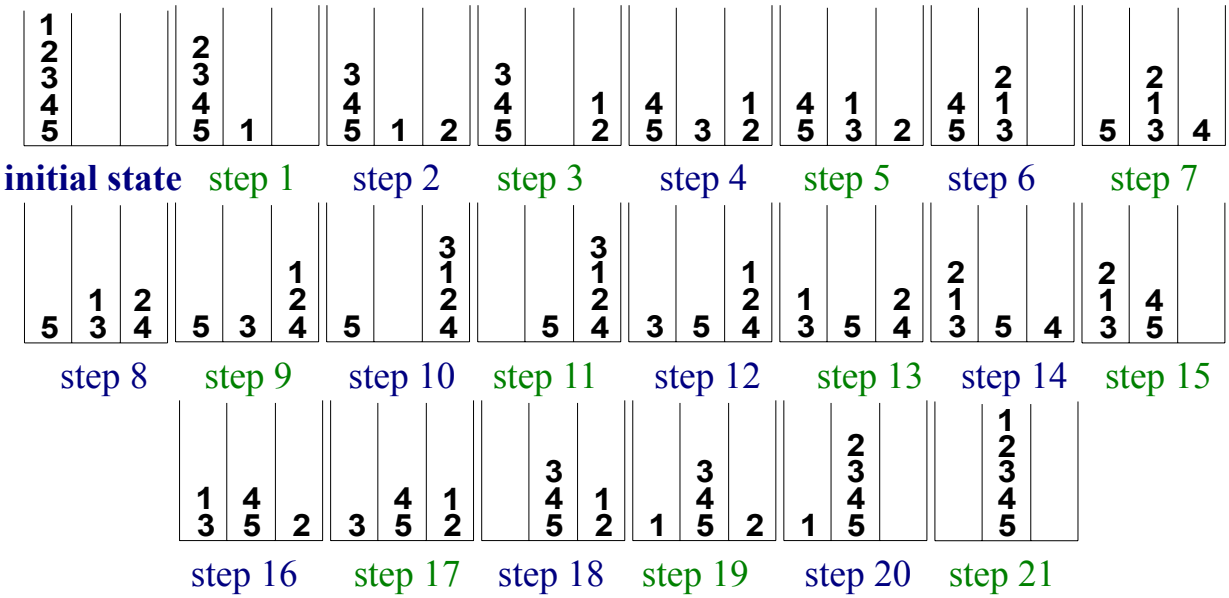


Fig. 1(The operation of puzzle with 5 disks)

In Fig.1, the numbers represent the sizes of disks, and the frames represent pegs.

The number of steps moving 5 disks in the traditional problem of Tower of Hanoi is 31, quite different from the number of steps in our puzzle.

2. Algorithm and Combinatorial Discussion

(1) Algorithm

Based on the above observation, the following algorithm achieving the minimum number of steps solving the puzzle can be derived; the time complexity is $O(n^2)$.

---

a lá Knuth Algorithm

1. ***For i ← 1 to n-1***
2. ***Move disk(i) to an empty peg(d)***
3. ***While disk(j) < disk(i)***
4. ***Move disk(i) to peg(d)***
5. ***Next***
6. ***Move disk(n) to the empty peg***
7. ***Move the disk(1) to disk(n-1) by the reverse operation***

---

(2) The Minimum Number of Steps Moving All Disks

Lemma 1

Suppose that n disks are placed on one peg in order of size. Let b_n be the minimum number of steps moving n disks to another peg according to Knuth's rules, i.e., only the biggest disk is required placed on the bottom while the remaining disks may be placed in any order. Let a_n be the minimum number of steps moving n disks to another peg with disks be placed in order of size at the end. Then we have the recurrence relations:

$$b_n = b_{n-1} + n,\ \ b_1 = 1$$

$$a_n = 2b_{n-1} + 1$$

■

Proof

According to the operation of the puzzle, we can derive:

$$b_n = b_{n-1} + (n-1) + 1 = b_{n-1} + n$$

$$a_n = b_{n-1} + 1 + b_{n-1} = 2b_{n-1} + 1$$

■

Theorem 1

The minimum number of steps moving n disks from one peg to another peg is given by:

$$a_n = n^2 - n + 1 = 2C_2^n + 1$$

■

Proof

Solve the recurrence relation in Lemma 1:

$$\begin{aligned} a_n &= 2b_{n-1} + 1 \\ &= 2[b_{n-2} + (n-1)] + 1 \\ &= a_{n-1} + 2n - 2 \\ a_n &= n^2 - n + 1 = 2C_2^n + 1 \end{aligned}$$

■

The number of steps moving n disks in the traditional Tower of Hanoi problem is $2^n - 1$, totally different from our modified game (n^2-n+1). Next, we discuss the mathematical properties in the puzzle.

(3) The Sequences of Recording the Disk Moves

In view of the symmetric properties observed in Table 1, we derive the following formula:

Formula 1

In the sequence $L_n(k)$, the disk x is moved for the first time at step P_x is given by:

$$P_x = \frac{x^2 - x + 2}{2} = C_2^x + 1$$ ■

Proof

We derive it from the symmetric property in $L_n(k)$.

$$1+2+3+\cdots\cdots+(x-1)+1 = \frac{x(x-1)}{2}+1$$ ■

According to symmetric property in the puzzle, we notice that disk x is moved for the first time is a **symmetry point** in $L_n(k)$ (The red numbers in Table 1), this property is important for us to design algorithms and solve the problem of the closed formula.

Table 1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
L_2	**1**	**2**	1																		
L_3	**1**	**2**	1	**3**	1	2															
L_4	**1**	**2**	1	**3**	1	2	**4**	2	1	3	1	2	1								
L_5	**1**	**2**	1	**3**	1	2	**4**	2	1	3	**5**	3	1	2	4	2	1	3	1	2	1

According to Formula 1 and symmetric property, we derive the non-recursive algorithm of the sequence of L_n. Its time complexity is $O(n^2)$.

Algorithm L_n

1. $L(m,1) \leftarrow 1$
1. *if* $n > \frac{m^2 - m + 2}{2}$ *then* $n \leftarrow m^2 - m + 2 - n$
2. *for* $i = 1$ *to* n {
3. $L(m, \frac{i^2 - i + 2}{2}) \leftarrow i$
4. *for* $j = i - 1$ *to* 1 *step* -1
5. $L(m, \frac{i^2 - i + 2}{2} + j) \leftarrow L(m, \frac{i^2 - i + 2}{2} - j)$
6. }

The recursive algorithm of $L_n(k)$; its time complexity is $O(n^2)$:

Algorithm $L_n(k)$

1. *for* $i = 1$ *to* n {
2. *if* $\frac{i^2 - i + 2}{2} = n$ *then* {
3. *Return* i
4. *exit for*
5. *else if* $\frac{i^2 - i + 2}{2} > n$ *then*
6. $symmetry \leftarrow i - 1$
7. *exit for*
8. }
9. $n \leftarrow (n - (n - \frac{symmetry^2 - symmetry + 2}{2}))$
10. }

(4) The Closed Formula

Although $L_n(k)$ is formed recursively, we are able to obtain a closed formula. Here $[a]$ stands for the biggest integer less than or equal to a.

Formula 2

$$\boldsymbol{L_n(k) = |x - 2i| + \left[\frac{i}{\left[\frac{x+1}{2}\right]}\right],\ x = \left[\frac{-1 + \sqrt{1 + 8(k-1)}}{2}\right] + 1,\ i = k - P_x,\ P_x = C_2^x + 1}$$ ■

Our method to get $L_n(k)$ has three parts:

1) Find the biggest disk x be moved before step k, x is a symmetry point.
2) Use sequence $T(x,i)$, which equals to the terms after disk x. We can prove it based on symmetry property in the puzzle.
3) $i = k\text{-}P_x$, we can get $L_n(k)$ through $T(x,i)$.

Lemma 2

$$x = \left[\frac{-1 + \sqrt{1 + 8(k-1)}}{2}\right] + 1$$ ■

Proof

$$P_x \le k < P_{x+1}$$

$$\frac{x^2 - x + 2}{2} \le k < \frac{x^2 + x + 2}{2}$$

$$\begin{cases} \frac{1 - \sqrt{1 + 8(k-1)}}{2} \le k \le \frac{1 + \sqrt{1 + 8(k-1)}}{2} \\ k > \frac{-1 + \sqrt{1 + 8(k-1)}}{2} \cup k < \frac{-1 - \sqrt{1 + 8(k-1)}}{2} \end{cases}$$

$$\frac{-1 + \sqrt{1 + 8(k-1)}}{2} < k \le \frac{1 + \sqrt{1 + 8(k-1)}}{2}$$

$$x=\left\lfloor \frac{-1+\sqrt{1+8(k-1)}}{2} \right\rfloor +1$$ ■

Lemma 3

$T(x,i)$ as:

x-2, x-4, x-6,……,1,2,4,6,……, x-1, when x is odd;

x-2, x-4, x-6,……,2,1,3,5,……, x-1,when x is even.

$$T(x,i)=\left|x-2i\right|+\left\lfloor \frac{i}{\left\lceil \frac{x+1}{2} \right\rceil} \right\rfloor$$ ■

Proof (Formula 2)

Based on Formula 1, Lemma 2 ,and Lemma 3, we can obtain the closed formula of $L_n(k)$. ■

3. Partition of Positive Integers

As indicated in Table 2, $L_n(k)$ forms a partition of positive integers (The red numbers):

Table 2

(The operation of 10 disks)

k / *s*	1	2	3	4	5	6	7	8	9	10
1	*1*	*2*	*4*	*7*	*11*	*16*	*22*	*29*	*37*	*46*
2	*3*	*6*	*10*	*15*	*21*	*28*	*36*	*45*	*55*	
3	*5*	*8*	*12*	*17*	*23*	*30*	*38*	*47*		
4	*9*	*14*	*20*	*27*	*35*	*44*	*54*	63		
5	*13*	*18*	*24*	*31*	*39*	*48*	56			
6	*19*	*26*	*34*	*43*	*53*	62	70			
7	*25*	*32*	*40*	*49*	57	64				
8	*33*	*42*	*52*	61	49	76				
9	*41*	*50*	58	65	71					
10	*51*	60	68	75	81					
11	59	66	72	77						
12	67	74	80	85						
13	73	78	82							
14	79	84	88							
15	83	86								
16	87	90								
17	89									
18	91									

In Table 2, k represents the number of disks; s represents the time that disk be moved.

(1) Doubly Periodicity

Table 3

(The partition of positive integers)

1	2	4	7	11	**16**	22	29	37	46
3	6	10	15	21	**28**	36	45	55	66
5	8	12	17	23	**30**	38	47	57	68
9	14	20	27	35	**44**	54	65	77	90
13	18	24	31	39	**48**	58	69	81	94
19	26	34	43	53	**64**	76	89	103	118
25	32	40	49	59	**70**	82	95	109	124
33	42	52	63	75	**88**	102	117	133	150
41	50	60	71	83	**96**	110	125	141	158
51	62	74	87	101	**116**	132	149	167	186
61	72	84	97	111	**126**	142	159	177	196
73	86	100	115	131	**148**	166	185	205	226
85	98	112	127	143	**160**	178	197	217	238
99	114	130	147	165	**184**	204	225	247	270
113	128	144	161	179	**198**	218	239	261	284
129	**146**	**164**	**183**	**203**	**224**	**246**	**269**	**293**	**318**
145	162	180	199	219	**240**	262	285	309	334
163	182	202	223	245	**268**	292	317	343	370
181	200	220	241	263	**286**	310	335	361	388
201	222	244	267	291	**316**	342	369	397	426

Table 4

(The result of modulo 5)

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

Table 3 becomes Table 4 when the numbers are taken modulo 5.

A striking doubly periodicity thus displayed:

1) If the numbers are read along a fixed column, they repeat themselves every 10 entries.
2) If the numbers are read along a fixed row, they repeat themselves every 5 entries.

Similar phenomena occur when the number 5 is replaced by any integer greater than 1. This interesting property is summarized by the following theorem.

Theorem 2

Let p be a positive integer greater than* 1.** ***In the table displaying the partition of positive integers associated with the Tower of Hanoi a lá Knuth with all entries reduced by taking modulo p, every column in* Table 3 *has a period* 2p, *and every row in* Table3 *has a period p.

The proof of this theorem depends on the following explicit formula of g(k,s), the step number when disk k is moved the s^{th} time in the puzzle:

1. $$g(k,s)=\frac{(s+k-1)(s+k-2)}{2}+1+\left[\frac{s+k-1}{2}\right]+(-1)^s\left[\frac{k}{2}\right]$$

2. $$g(k,s)=\frac{s(s-1)}{2}+\left[\frac{s}{2}\right]+1+\frac{(2s+k-2)(k-1)}{2}+\frac{(k-1)\left[1+(-1)^s\right]}{2}$$

■

Lemma 4

$$f(x) = ax^2 + bx + c \,, a, b, c \in Z$$
$$f(x) \equiv f(x+p) \pmod{p}$$

Proof

$$0 \equiv 2apx + ap^2 + bp \pmod{p}$$
$$ax^2 + bx + c \equiv (ax^2 + bx + c) + 2apx + ap2 + bp \pmod{p}$$
$$ax^2 + bx + c \equiv a(x+m)^2 + b(x+m) + c \pmod{p}$$
$$f(x) \equiv f(x+p) \pmod{p}$$

Proof (Theorem 2)

Discuss the formula of g(k,s):

$$s = 2r\text{-}1 : g(k,s) = 2r^2 - 2r + 1 + \frac{(4r-4+k)(k-1)}{2} = \frac{k^2 + (4r-5) + 4r^2 - 8r + 6}{2}$$
$$s = 2r : \quad g(k,s) = 2r^2 + 1 + \frac{(4r+k)(k-1)}{2} = \frac{k^2 + (4r-1)k + 4r^2 - 4r + 2}{2}$$

$g(k,s) \in \mathrm{N}$, and

$$k^2 + (4r-5) + 4r^2 - 8r + 6$$
$$k^2 + (4r-1)k + 4r^2 - 4r + 2$$

Both of them correspond to Lemma 4; every row has a period p.

When we prove the periodicity in every column by the same way, it would get $[\frac{k}{2}]$ or $[\frac{s}{2}]$ in g(k,s); every column has a period $2p$.

(2) The Closed Formula

g(k,s) is the step number when disk k is moved the s[th] time in the puzzle

Formula 3

1. $g(k,s) = \frac{(s+k-1)(s+k-2)}{2} + 1 + \left[\frac{s+k\text{-}1}{2}\right] + (-1)^s \left[\frac{k}{2}\right]$
2. $g(k,s) = \frac{s(s-1)}{2} + \left[\frac{s}{2}\right] + 1 + \frac{(2s+k-2)(k-1)}{2} + \frac{(k-1)\left[1+(-1)^s\right]}{2}$

In order to obtain these formulas, we have to prove several lemmas.

Lemma 5

$$x = k + s - 1$$

Divide $L_n(k)$ into groups according to the position of every symmetry point, as in Fig. 2. Every group has x terms.

We can prove it through symmetric property in $L_n(k)$.

$L_n(k)$ 1 21 312 4213 53124 642135 7531246 8642135 9…

Fig. 2 (The grouping of $L_n(k)$)

Lemma 6

$T^{-1}(x,m)=i$ is the inverse function of $T(x,i)=m$.

$$T^{-1}(x,m)=\begin{cases}[\frac{x}{2}]-[\frac{m}{2}], x\equiv m \pmod 2\\ [\frac{x}{2}]+[\frac{m}{2}], x\equiv m+1 \pmod 2\end{cases}$$

Formula 3.1

$$g(k,s)=\frac{(s+k-1)(s+k-2)}{2}+1+\left[\frac{s+k-1}{2}\right]+(-1)^s\left[\frac{k}{2}\right]$$

Proof

According to Lemma5, Lemma 6, and Formula 1:

$$g(k,s)=P_{k+s-1}+T^{-1}(k+s-1,k)$$
$$=\frac{(s+k-1)(s+k-2)}{2}+1+\left[\frac{s+k-1}{2}\right]+(-1)^s\left[\frac{k}{2}\right]$$

Then, we would like to get recurrence relation of $g(k,s)$.

Listing the difference of any two columns in Table 3, we get Table 5:

Table 5

(Difference of any two columns in partition of positive integers)

1	2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10	11
3	4	5	6	7	8	9	10	11
5	6	7	8	9	10	11	12	13
5	6	7	8	9	10	11	12	13
7	8	9	10	11	12	13	14	15
7	8	9	10	11	12	13	14	15
9	10	11	12	13	14	15	16	17
9	10	11	12	13	14	15	16	17
11	12	13	14	15	16	17	18	19
11	12	13	14	15	16	17	18	19
13	14	15	16	17	18	19	20	21
13	14	15	16	17	18	19	20	21
15	16	17	18	19	20	21	22	23
15	16	17	18	19	20	21	22	23
17	18	19	20	21	22	23	24	25
17	18	19	20	21	22	23	24	25
19	20	21	22	23	24	25	26	27
19	20	21	22	23	24	25	26	27
21	22	23	24	25	26	27	28	29

In Table 5, we conjecture this regularity:

Row 2r = Row 2r-1 +2

Row 2r = Row 2r+1

To prove this conjecture, we want to Lemma 7:

Lemma 7

Let $k_{s\to s+1}$ be the difference of disk k is moved the s^{th} time and $(s+1)^{th}$ time.

x is symmetry point.

$$(k+1)_{s\to s+1}-k_{s\to s+1}=\begin{cases}2, x\equiv k \pmod 2\\ 0, x\equiv k+1 \pmod 2\end{cases}$$

■

Proof

The regularity in Table 5 can be summarized by this mathematical result. Because:

$$\begin{aligned}&[g(k+1,s+1)-g(k,s+1)]-[g(k+1,s)-g(k,s)]\\ =&[g(k+1,s+1)-g(k+1,s)]-[g(k,s+1)-g(k,s)]\\ =&(k+1)_{s\to s+1}-k_{s\to s+1}\end{aligned}$$

Then we calculate two condition of Lemma 7 in $L_n(k)$.

1) $x\equiv k \pmod 2$

$L_n(k)$: $\cdots\cdots\underline{x\cdots\cdots k\cdots k+1\cdots}\quad\underline{x+1\cdots k+1\cdots k\cdots\cdots}\quad\underline{x+2\cdots\cdots\cdots k\cdots k+1\cdots}$

Times $\qquad\qquad\qquad s\quad s\text{-}1\qquad\qquad\qquad s\quad s+1\qquad\qquad\qquad\qquad s+2\quad s+1$

Fig. 3

$$\begin{aligned}k_{s\to s+1}&=x+T^{-1}(x+1,k)+1-T^{-1}(x,k)-1\\ (k+1)_{s\to s+1}&=(x+1)+T^{-1}(x+2,k+1)+1-T^{-1}(x+1,k+1)-1\\ (k+1)_{s\to s+1}-k_{s\to s+1}&=(x+1+\left[\frac{x+2}{2}\right]+\left[\frac{k+1}{2}\right]+1-\left[\frac{x+1}{2}\right]+\left[\frac{k+1}{2}\right]-1)\\ &\quad-(x+\left[\frac{x+1}{2}\right]+\left[\frac{k}{2}\right]+1-\left[\frac{x}{2}\right]+\left[\frac{k}{2}\right]-1)\\ &=\left[\frac{x+2}{2}\right]+2\left[\frac{k+1}{2}\right]-2\left[\frac{x+1}{2}\right]-2\left[\frac{k}{2}\right]+\left[\frac{x}{2}\right]+1\end{aligned}$$

$k\equiv 0 \pmod 2$

$$\begin{aligned}(k+1)_{s\to s+1}-k_{s\to s+1}&=\frac{x+2}{2}+k-x-k+\frac{x}{2}+1\\ &=2\end{aligned}$$

$k\equiv 1 \pmod 2$

$$\begin{aligned}(k+1)_{s\to s+1}-k_{s\to s+1}&=\frac{x+1}{2}+(k+1)-(x+1)-(k-1)+\frac{x-1}{2}+1\\ &=2\end{aligned}$$

2) $x \equiv k+1 (\mathrm{mod}\, 2)$

$\boldsymbol{L_n(k)}$: $\cdots\cdots\underline{x\cdots\cdots k{+}1\cdots k\cdots}$ $\underline{x{+}1\cdots k\cdots k{+}1\cdots\cdots}$ $\underline{x{+}2\cdots\cdots\cdots k{+}1\cdots k\cdots}$

Times $s\text{-}1$ s $s{+}1$ s $s{+}1$ $s{+}2$

Fig. 4

$$k_{s\to s+1} = x + T^{-1}(x+1,k) + 1 - T^{-1}(x,k) - 1$$

$$(k+1)_{s\to s+1} = (x+1) + T^{-1}(x+2,k+1) + 1 - T^{-1}(x+1,k+1) - 1$$

$$\begin{aligned}(k+1)_{s\to s+1} - k_{s\to s+1} &= (x+1+\left[\frac{x+2}{2}\right]-\left[\frac{k+1}{2}\right]+1-\left[\frac{x+1}{2}\right]-\left[\frac{k+1}{2}\right]-1)\\ &\quad -(x+\left[\frac{x+1}{2}\right]-\left[\frac{k}{2}\right]+1-\left[\frac{x}{2}\right]-\left[\frac{k}{2}\right]-1)\\ &=\left[\frac{x+2}{2}\right]-2\left[\frac{x+1}{2}\right]-2\left[\frac{k+1}{2}\right]+2\left[\frac{k}{2}\right]+\left[\frac{x}{2}\right]+1\end{aligned}$$

$k \equiv 0 (\mathrm{mod}\, 2)$

$$\begin{aligned}(k+1)_{s\to s+1} - k_{s\to s+1} &= \frac{x+1}{2} - (x+1) - k + k + \frac{x-1}{2} + 1\\ &= 0\end{aligned}$$

$k \equiv 1 (\mathrm{mod}\, 2)$

$$\begin{aligned}(k+1)_{s\to s+1} - k_{s\to s+1} &= \frac{x+2}{2} - x - (k+1) + (k-1) + \frac{x}{2} + 1\\ &= 0\end{aligned}$$

■

Formula 3.2

$$g(k,s) = \frac{s(s-1)}{2} + \left[\frac{s}{2}\right] + 1 + \frac{(2s+k-2)(k-1)}{2} + \frac{(k-1)\left[1+(-1)^s\right]}{2}$$

■

Proof

First, we calculate g(1,s):

$$\begin{aligned}g(1,s) &= P_s + T^{-1}(s,1)\\ &= \frac{s(s-1)}{2} + [\frac{s}{2}] + 1\end{aligned}$$

According to formula2, Lemma 7, and the formula $g(1,s)$, we derive following recurrence relation:

$$g(k,s) = g(k-1,s) + s + (k-2) + \frac{1+(-1)^s}{2}$$

Solving this recursion, we derive:

$$g(k,s) = \frac{s(s-1)}{2} + \left[\frac{s}{2}\right] + 1 + \frac{(2s+k-2)(k-1)}{2} + \frac{(k-1)\left[1+(-1)^s\right]}{2}$$

■

4. The Ordering of Proper Fractions associated with the Fibonacci Numbers

We examine this interesting phenomenon:

Here is the set of all proper fractions, listed in the order of size, with both the denominator and numerator taken from the first 8 terms of the Fibonacci numbers 1, 1, 2, 3, 5, 8, 13, 21:

$$\underset{1}{\frac{1}{21}}<\underset{2}{\frac{1}{13}}<\underset{3}{\frac{2}{21}}<\underset{4}{\frac{1}{8}}<\underset{5}{\frac{3}{21}}<\underset{6}{\frac{2}{13}}<\underset{7}{\frac{1}{5}}<\underset{8}{\frac{3}{13}}<\underset{9}{\frac{5}{21}}<\underset{10}{\frac{2}{8}}<\underset{11}{\frac{1}{3}}<\underset{12}{\frac{3}{8}}<\underset{13}{\frac{8}{21}}<\underset{14}{\frac{5}{13}}<\underset{15}{\frac{2}{5}}<\underset{16}{\frac{1}{2}}<\underset{17}{\frac{3}{5}}<\underset{18}{\frac{8}{13}}<\underset{19}{\frac{13}{21}}<\underset{20}{\frac{5}{8}}<\underset{21}{\frac{2}{3}}$$

Looking at the fractions with a fixed denominator, say 21: $\frac{1}{21},\frac{2}{21},\frac{3}{21},\frac{5}{21},\frac{8}{21},\frac{13}{21}$, we note that they occur at places 1, 3, 5, 9, 13, 19, and these are precisely the numbers in the first column of Table 3. Similarly, if we look at the fractions with a fixed numerator, say 1: $\frac{1}{21},\frac{1}{13},\frac{1}{8},\frac{1}{5},\frac{1}{3},\frac{1}{2}$, we note that they occur at places 1, 2, 4, 7, 11, 16, and these are precisely the numbers in the first row of **Table 3**. Was this an accident, or some kind of rule is followed?

Further investigation takes us to this theorem.

***Theorem* 3**

The column/row of the partition table is exactly the ordering sorting the proper fractions associated with the Fibonacci numbers of fixed denominator/numerator. ■

Proof

We prove this theorem by combinatorial proof.

We construct $L_n(k)$ by the symmetric property in the puzzle, and construct the order of proper fractions associated the Fibonacci numbers by Lemma 8. The methods of constructing them are the same. ■

(1) 5
(2) 454
(3) 3435343
(4) 23242353242 32
(5) 12131242135312421312 1

Fig. 5
(The method of constructing $L_n(k)$)

(1) $\frac{0}{1}<\frac{1}{1}$

(2) $\frac{0}{1}<\frac{1}{2}<\frac{1}{1}$

(3) $\frac{0}{1}<\frac{1}{3}<\frac{1}{2}<\frac{2}{3}<\frac{1}{1}$

(4) $\frac{0}{2}<\frac{1}{5}<\frac{1}{3}<\frac{2}{5}<\frac{1}{2}<\frac{3}{5}<\frac{2}{3}<\frac{1}{1}$

(5) $\frac{0}{3}<\frac{1}{8}<\frac{1}{5}<\frac{2}{8}<\frac{1}{3}<\frac{3}{8}<\frac{2}{5}<\frac{1}{2}<\frac{3}{5}<\frac{5}{8}<\frac{2}{3}<\frac{1}{1}$

Fig. 6
(The order of fraction associated with $\mathbf{F_n}$)

Lemma 8

$$a,b,c,d \in \mathrm{N}\,.\, a > b, c > d\,, \frac{b}{a} < \frac{d}{c} \quad \text{then} \frac{b}{a} < \frac{b+d}{a+c} < \frac{d}{c}$$ ■

Proof

$$ab + bc < ab + ad \rightarrow \frac{b}{a} < \frac{b+d}{a+c}$$

$$cd + bc < cd + ad \rightarrow \frac{b+d}{a+c} < \frac{d}{c}$$

$$\frac{b}{a} < \frac{b+d}{a+c} < \frac{d}{c}$$ ■

5. Algorithm of Restoration

Paper [4] offers an algorithm for restoring back to the initial state from any state in the Tower of Hanoi problem. Its secret is best explained with the state-space graph in which every point represents a state in Tower of Hanoi. In order to approach the similar problem for the Tower of Hanoi a lá Knuth, we have designed a Rug Paving Algorithm. It turns out that the procedures thus produced correspond to certain paths in a similar state graph.

(1) Rug Paving Algorithm

According to symmetry and mathematical properties shown in the puzzle, we are able to design the Rug Paving Algorithm for restoring back to initial state. Its time complexity is $\boldsymbol{O(n^2)}$.

There is no guarantee such an algorithm takes the minimum number of steps. According to the quantity of all states, if there is an algorithm takes the minimum number of steps, its time complexity is $\boldsymbol{O(n^n)}$. Therefore, Rug Paving Algorithm can solve all states in the puzzle.

Suppose that we have n disks. Our algorithm has three parts:

Part I (Divide Algorithm):

Our goal is to move the disks so at the end disks n, n-1 and n-2 are each placed on the bottom of separate pegs. To this end, we need to identify disks, which are **visible from the top,** each of which is no smaller than disks lying above. By deleting disks lying below a visible disk, a new initial state is formed. Applying recursion to all disks **S** formed by deleting all disks below the second biggest disk, we may move **S** to an initially empty peg. Disk n-2 can now be separated regardless of its initial position.

Part II (Unrestricted Algorithm):

Leaving disks n, n-1, n-2 fixed, move the rest so at the end disks 1, 2,⋯, n-2 are placed in one single peg in the correct order.

Part III (a lá Knuth Algorithm):

To complete the restorations, it remains to move the stacks according to the Tower of Hanoi a lá Knuth.

Divide Algorithm

```
1.  ring[0...2][0...n]
2.  k[0...2]
3.  count←1
4.  point[0...n]
5.  max←0
6.  for i = 0 to 2{
7.      if (ring[i][1]=1){
8.          k[0]←i
9.      else
10.         k[count]←i
11.         count←count+1
12.     }
13.  }
14.  for i = 1 to 2{
15.      for j = len(ring[k[i]]) down to 0
16.          MOVE(ring[k[i]][j],k[0])
17.  }
18.  count←0
19.  for j =2 to len(ring[k[0]]){
20.     i ←j
21.     while i < len(ring[k[0]]) do
22.         max←Max(max,ring[k[0]][i])
23.         i ← i +1
24.     point[count]←max
25.      count←count+1
26.      max←0
27.  }
28.  i←1
29.   count←count-1
30.  while(ring[k[1]][0] × ring[k[2]][0]!=2) do{
31.     for j = len(ring[k[0]]) down to point[count]
32.         MOVE(ring[k[0][j]],i)
33.  }
34.  i←(i+1) mod 2
35.   count←count-1
```

Unrestricted Algorithm

```
1.  for i =1 to 2{
2.     for j = len(ring[k[i]]) down to 0
3.         MOVE(ring[k[i]][j],k[0])
4.  }
5.  count←0
6.  for i = n down to 1{
```

```
7.      for j = len(ring[k[0]]) down to demand[count]
8.          MOVE(ring[k[0]][i],k[2])
9.      MOVE(ring[k[0]][j],k[1])
10.     for j = len(ring[k[0]]) down to 0
11.         MOVE(ring[k[0]][j],k[1])
12.     count←count-1
13. }
```

---

a lá Knuth algorithm

```
1.  for i = 1 to n-1
2.      MOVE(demand[count] ,demandcolumn)
3.  while ring[k[0]][i]<demand[count]
4.      MOVE(ring[k[0]][i],demandcolumn)
5.  MOVE (n,1)
6.  for i = n-1 down to 1{
7.      while ring[k[0]][i]<demand[count]
8.          MOVE(ring[k[0]][i],demandcolumn)
9.          MOVE(demand[count] ,demandcolumn)
10. }
```

---

(2) State Graph

We declare that the puzzle is in the state of (*i, j, k)* when pegs A, B, C each contains *i, j, k* disks respectively. The states of the puzzle can now be illustrated as in **Fig. 4**. We are able to chart the moves as paths in this graph. The decision for moving the next disk corresponds to moving in this graph from one node to its neighbor. The number of steps in applying the Divide Algorithm can be reduced by taking this as a visual guide. Similarly, the graph can serve as a visual guide in applying the "a lá Knuth Algorithm".

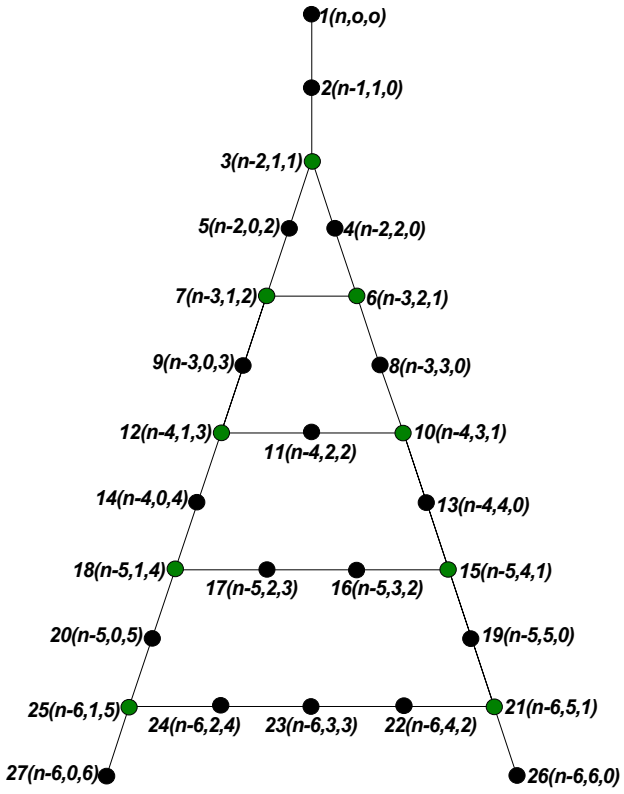


Fig. 7(State Graph)

State Graph Algorithm

```
1.  struct State Graph{
2.      int a
3.      int b
4.      int c
5.  }
6.  t ← 0
7.  for i = 1 to n-1{
8.      t = t + 1
9.      MOVE(demand[count] ,demandcolumn)
10.     State Graph[t], A←len(ring[k[0]])
11.     State Graph[t], B←len(ring[k[1]])
12.     State Graph[t], C←len(ring[k[2]])
13. }
14. while ring[k[0]][i]<demand[count]{
15.     t = t + 1
16.     MOVE(ring[k[0]][i],demandcolumn)
17.     State Graph[t], A←len(ring[k[0]])
18.     State Graph[t], B←len(ring[k[1]])
19.     State Graph[t], C←len(ring[k[2]])
20. }
21. t = t + 1
22. MOVE (n,1)
23. State Graph[t], A←len(ring[k[0]])
24. State Graph[t], B←len(ring[k[1]])
25. State Graph[t], C←len(ring[k[2]])
26. for i = n-1 down to 1{
27.     while ring[k[0]][i]<demand[count]{
28.         t = t + 1
29.         MOVE(ring[k[0]][i],demandcolumn)
30.         State Graph[t], A←len(ring[k[0]])
31.         State Graph[t], B←len(ring[k[1]])
32.         State Graph[t], C←len(ring[k[2]])
33.         t = t + 1
34.         MOVE(demand[count] ,demandcolumn)
35.         State Graph[t], A←len(ring[k[0]])
36.         State Graph[t], B←len(ring[k[1]])
37.         State Graph[t], C←len(ring[k[2]])
38.     }
39. }
```

Summary

In this project, we have explored the Puzzle of Tower of Hanoi a lá Knuth. The theme is focused on the Computer Algorithm, Number theory, Combinatorics. We have:

1. In the modified puzzle, the minimum number of steps moving n disks is n^2-n+1, totally different from 2^n-1, the number of steps of the traditional Puzzle of Tower of Hanoi.

2. This puzzle defines a partition of the positive integers and the associated table is doubly periodic mod p.

3. We prove that the order of proper fraction associated with Fibonacci numbers is exactly the partition formed by this puzzle.

4. We devise an algorithm to restore the puzzle from an arbitrary state back to the "all visible from the top" state.

5. In terms of complexity, the traditional Tower of Hanoi requires 2^n-1 steps, while the modified puzzle requires n^2-n+1 steps. It would be interesting to see if any other modified puzzles having the degree of hardness lying "in between". We conjecture that the intermediate games are realized for puzzles $G(k)$ obeying the rule that the k biggest ones be at the bottom, $k \geq 2$. It would be interesting to see if there are recursive relations among $G(k)$'s.

References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithm*, II, MIT Press, 123-195, 2001.

[2] R. Graham, D. Knuth and O. Patashnik, *Concrete Mathematics*, Addison-Wesley, 1989.

[3] D. Knuth, All questions answered, *Notices A.M.S.* **49**(3), 318-324, 2002.

[4] D. G. Poole, The Towers and Triangle of Professor Claus (or, Pascal Knows Hanoi), *Mathematics Magazine*, **67**(5), 323-343, 1994.

[5] Problem #10956, *Amer. Math. Monthly*, Proposed by J. McCarthy, 109(7), 664, Aug.-Sept. 2002, Solution by M. Stamp (Preprint).

評語及建議事項

此作品極具原創性，作者科學精神與態度佳，思考程序完整，作品具學術性。作者可再多思考其作品的應用層面，在作品呈現上，作者可再調整，建議如下：一、描述問題；二、所提演算法；三、演算法正確性之證明；四、演算法複雜度之分析；五、演算法的應用。