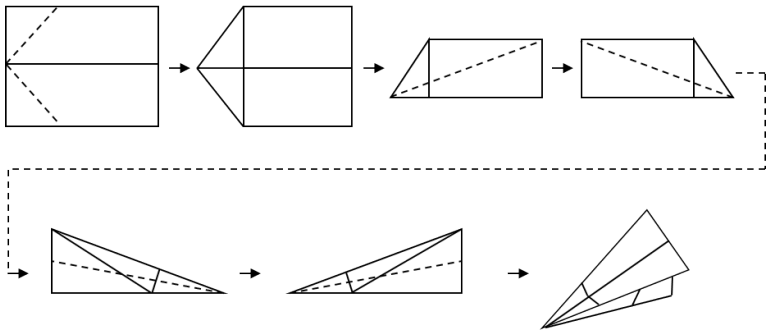
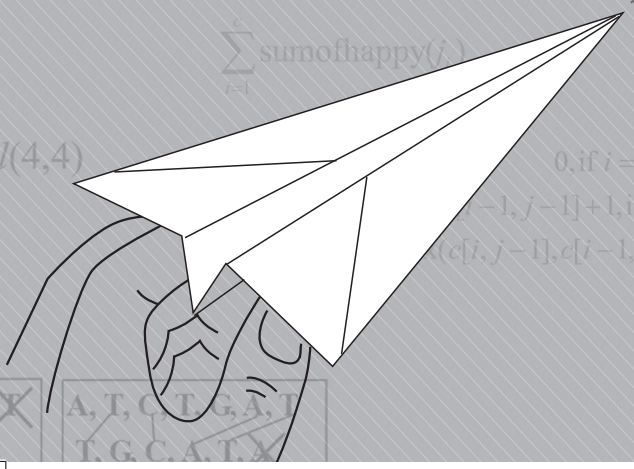
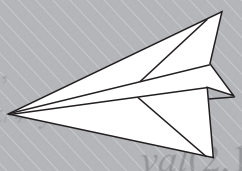
1 一切從觀察開始…

章節大綱



摺紙飛機的過程隱含一個演算法





1.1 何謂演算法？

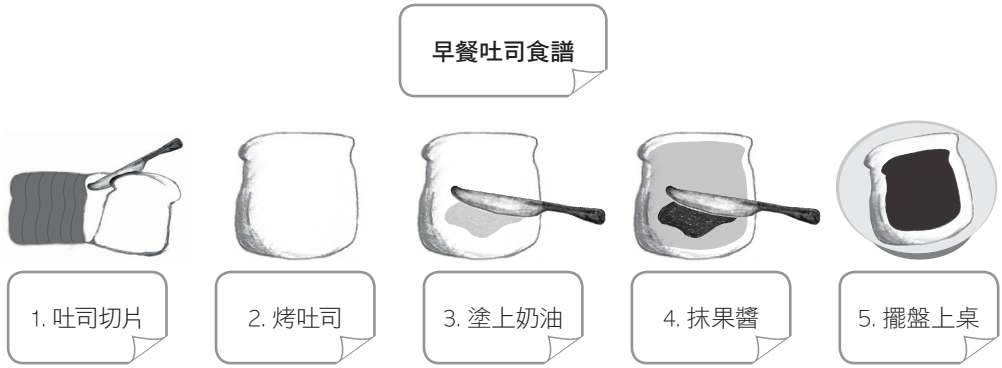
「何謂演算法(algorithms)?」

「簡言之，一個演算法是在符合問題的限制下，將輸入(input)轉換成輸出(output)的過程。」

電腦演算法是人類利用電腦解決問題的技巧之一。其實每個人都會一些演算法。例如，廚師會將食材轉換成美食；演奏家會將樂譜轉換成迷人的音樂；甚至小朋友將廢紙摺成紙飛機的過程也隱含一個演算法。而程式設計師(programmer)就是利用電腦來執行某一個演算法，以解決特定問題的人。

表 1.1 演算法存在於各行業中

執行者	輸入	輸出
廚師	食材	美食
演奏家	樂譜	音樂
小朋友	廢紙	紙飛機
程式設計師	程式輸入	程式輸出



「如何設計演算法？」

設計演算法的第一個好習慣是「觀察」。我們認為「觀察是一切發現的開端」。本書的開始，利用一個例子：河內塔(Hanoi tower)問題，來說明「觀察」和「設計演算法」的關係。

1.2 河內塔問題

河內塔問題是一個古老的遊戲。遊戲的目的是：將左方柱子上的碟子搬到右方的柱子。遊戲的規則有三條：

(1) 一次搬一個碟子，

(2) 每根柱子只有最上面的碟子可被搬動，

(3) 大碟子不可置於小碟子的上方。

任何人可以找到三個碟子河內塔問題的一個解法，如下圖所示。

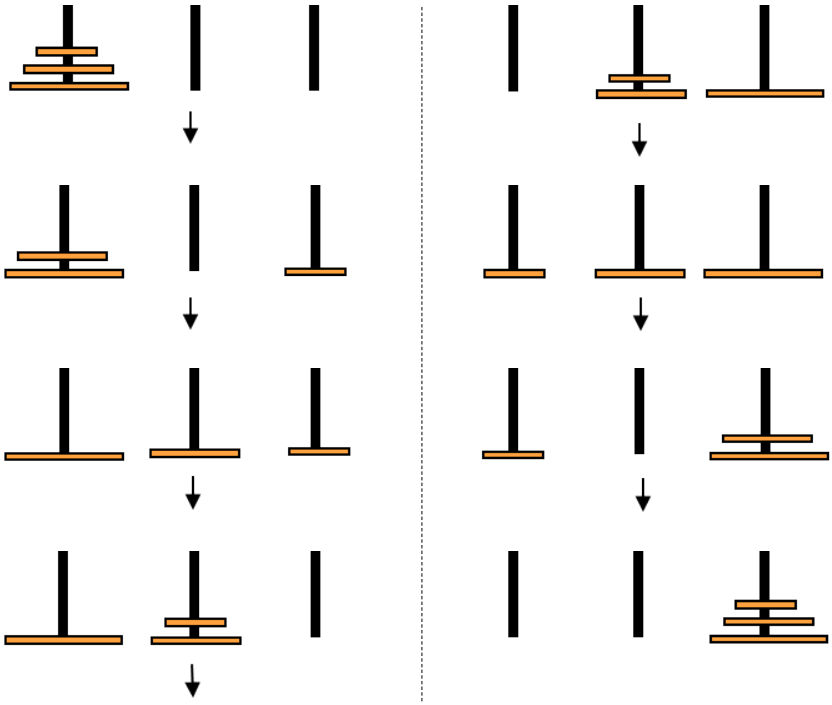


圖 1.1 三個碟子河內塔問題的一個解法

從上圖得知，三個碟子的河內塔問題共需要七個步驟。很自然地，我們提出的第一個問題是：

「n 個碟子的河內塔問題，共需要幾個步驟完成？」

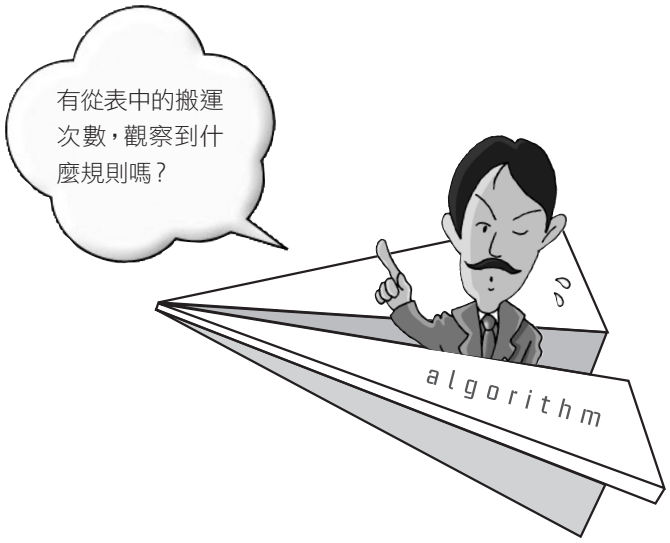
若這個問題無法立刻回答，可先觀察一些範例。

「觀察一些小例子，並記錄其移動的次數。」

為記錄方便，我們使用數學符號 T_n 來代表 n 個碟子河內塔問題所需要的搬運次數。並嘗試算出 n 小於 9 之前的 T_n，如表 1.2。

表 1.2 觀察河內塔問題的搬運次數

T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
0	1	3	7	15	31	63	127	255	?



「這個數列有何規則？T_9=？」

「T_n 好像是很靠近 2 的幾次方？T_n 是 2^n-1 嗎？」

「如何證明 T_n 是 2^n-1？」

「不知道！」

「還有其他規則？」

「比較前後項的關係，好像後項是前項的兩倍？不！是兩倍加 1。」

「可以使用符號，將此關係寫下來嗎？」

「可以！$T_n=2T_{n-1}+1$。」

「為何前後項有這樣的關係？」

「不知道！」

「如果 $T_n=2T_{n-1}+1$ 是正確的，有助於證明 $T_n=2^n-1$ 嗎？」「最常用的證明方法為何？」

「也許可以利用數學歸納法(mathematical induction)來證明 $T_n=2^n-1$。」

以下即是如此的證明。

表 1.3 利用數學歸納法證明 $T_n=2^n-1$

證明 $T_n=2^n-1$，當 $n\geq 0$ 時
Basis step: $T_0=2^0-1=1-1=0$，成立。 **Inductive step:** 假設 $n<k$，$T_n=2^n-1$ 成立。 當 $n=k$，$T_k=2\times T_{k-1}+1$。 因為 $T_{k-1}=2^{k-1}-1$，故 $T_k=2\times(2^{k-1}-1)+1=2^k-1$，成立。

另外一個簡單的證明如下。

表 1.4 $T_n=2^n-1$ 的另外一個證明

證明 $T_n=2^n-1$，當 $n\geq 0$ 時
$T_0=0$ $T_n=2T_{n-1}+1$ $T_n+1=2T_{n-1}+2$ 令 $U_n=T_n+1$，則 $U_0=1$，$U_n=2U_{n-1}$，$U_n=2^n$ $T_n=U_n-1=2^n-1$。

我們只利用簡單的觀察，便猜中 $T_n=2^n-1$ 這個性質。雖然，尚不知為何 $T_n=2T_{n-1}+1$。但此性質有助於證明 $T_n=2^n-1$。接下一個問題是：

「給一個 n 個碟子的河內塔問題，如何(找出)搬動碟子(的演算法)？」

「若這個問題無法立刻回答，可先嘗試找出一個小例子的解答，並觀察這個解答有何規則。」

以下為四個碟子河內塔問題的一個解答。根據先前發現的性質，我們得知共需要 $2^4-1=15$ 次搬動，如下圖所示。

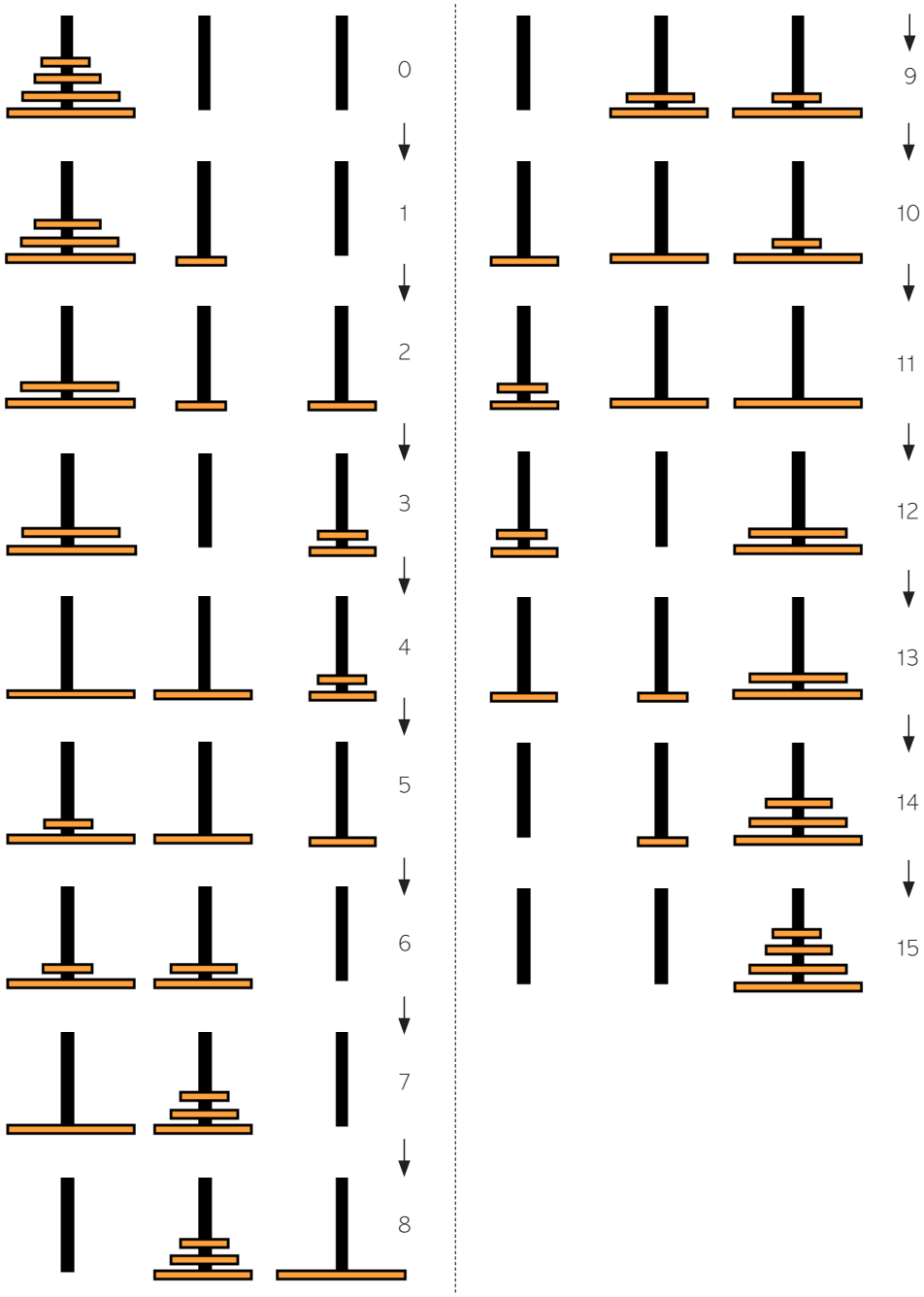


圖 1.2　四個碟子的河內塔問題之搬動過程

「搬動的過程中有何規則？」

「好像有些成堆的碟子打散之後又成堆。」

「可以解釋其中的道理嗎？」

「大概是為了搬動最底部且最大的碟子，必須先將上面的三個碟子全部搬到中間的柱子。」

「哪個步驟可以看得出此特性？」

「步驟 7。」

「一定需要這樣搬嗎？」

「應該沒有其他可能吧。倘若上面的任一個碟子散落於左邊的柱子，則最底部且最大的碟子不能被搬動；若上面的任一個碟子散落於右邊的柱子，則最底部且最大的碟子無處可置放。」

「為何是步驟 7？」

「因為前七個步驟是為了將上面的三個碟子全部搬到中間的柱子，第八步驟將最大的碟子搬到右邊的柱子，剩下的七個步驟是將先前置於中間的柱子的三個碟子再全部搬到右邊的柱子。」

「如此也說明 $T_4=2T_3+1$？」「倘若是 n 個碟子的河內塔問題，是否也可以利用搬 n-1 個碟子兩次來完成？」

答案是可以的。為搬動 n 個碟子的河內塔，可以先搬 n-1 個碟子到中間的柱子後，將最大的碟子搬到右邊的柱子，最後將中間的 n-1 個碟子搬到右邊的柱子。如此也說明先前發現的規律 $T_n=2T_{n-1}+1$，因為搬動 n 個碟子需要搬動 n-1 個碟子兩次及最大的碟子一次，如下圖所示。

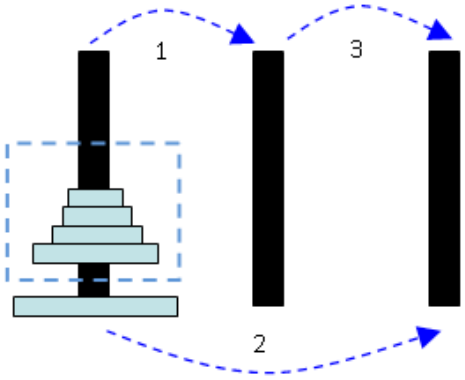


圖 1.3 解決五個碟子的河內塔問題需要搬動四個碟子兩次及最大碟子一次

根據上述的道理，我們可以設計下列的演算法來解決 n 個碟子的河內塔問題。

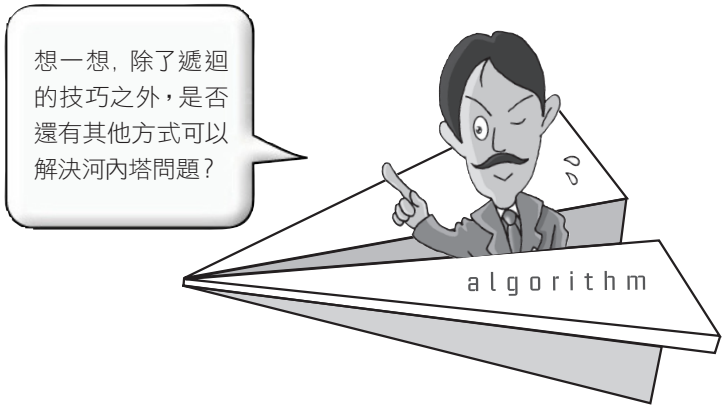
表 1.5 河內塔的遞迴演算法

輸入	在 *A* 柱的 *n* 個碟子
輸出	在 *C* 柱的 *n* 個碟子
步驟	

```
Algorithm TowersOfHanoi (n, A, C, B)
/* 搬動 n 個碟子從柱子 A 到柱子 C，可借用柱子 B*/
{
    if (n>=1) then
    {
        TowersOfHanoi(n-1, A, B, C);
        /* 從柱子 A 搬 n-1 個碟子到中間的 B 柱子 */

        write ("從柱子", A, "搬動一個碟子到柱子", C);
        TowersOfHanoi(n-1, B, C, A);
        /* 從柱子 B 搬 n-1 個碟子到中間的 C 柱子 */
    }
}
```

Tip 注意以上的副程式呼叫自己的概念，是一個很常使用的程式設計技巧「遞迴」(recursion)。



1.3 河內塔問題的非遞迴演算法

我們已經找到一個解決河內塔問題的遞迴演算法，另一個有趣的問題是：

「如果不用遞迴的演算法，可以解決河內塔問題嗎？」

「好像不行？」

「沒關係！先觀察幾個小例子。注意從輸入轉換成輸出的過程。並且用符號記錄整個移動過程。」

三個碟子的河內塔問題其搬動過程，可以記錄成下圖。為了方便，利用 1 到 3 的整數代表由小到大的三個碟子，而 ABC 代表三根柱子的位置。

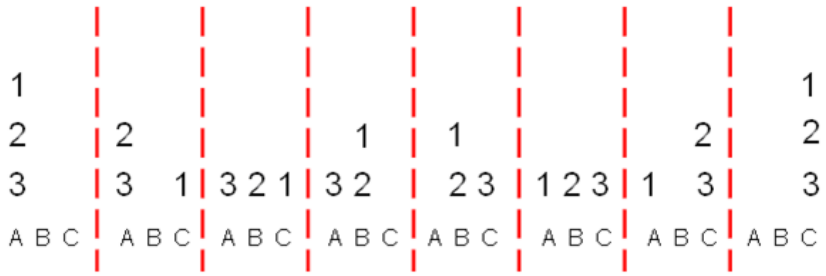


圖 1.4 記錄三個碟子河內塔問題的整個搬動過程

「看起來沒有規則？」

「不要放棄！嘗試另一種表示法。」

因為三個碟子河內塔問題共需要 2^3-1=7 次搬動。我們需要記錄 7 次搬動之中，每次的碟子號碼及搬動方向(即從哪一根柱子搬到哪一根柱子)，如下圖。

步驟	1	2	3	4	5	6	7
移動的碟子	1	2	1	3	1	2	1
移動的方向	A → C	A → B	C → B	A → C	B → A	B → C	A → C

圖 1.5 換一種方式記錄三個碟子的河內塔搬動過程

「還是看不出有什麼規則？」

「不要放棄！再嘗試另一種表示法。」

因為搬動的方向似乎不易觀察出規律。下圖將移動的方向標示的更簡潔一些。

步驟	1	2	3	4	5	6	7
移動的碟子	1	2	1	3	1	2	1
C	↑		↓	↑		↑	↑
B		↑	↓		↓		
A				↑			↑

圖 1.6 再換一種方式記錄三個碟子的河內塔搬動過程

「好像…？」

「也許應該試試 n 大一點的例子？」

以下是 n=4 的例子。

步驟	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
移動的碟子	1	2	1	3	1	2	1	4	1	2	1	3	1	2	1
C		↑	↑		↓	↓		↑	↑		↓	↑		↑	↑
B	↑	↓	↓	↑	↓	↓	↑	↑	↓	↓	↓	↓	↑	↑	↓
A	↑	↑		↑	↓		↑	↑		↓	↓		↑	↑	

圖 1.7　記錄四個碟子的河內塔搬動過程

「注視搬運的碟子號碼，是否找得到規律？」

「好像 1 號碟子出現在奇數的搬動次數上？」

「其他碟子也有這種現象嗎？」

「2 號碟子沒有這個現象。但是，好像每個 2 號碟子隔四次才搬動一次？」

「專心觀察 2 號碟子被搬動的步驟。」

步驟	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
移動的碟子		2				2				2				2	

圖 1.8　觀察 2 號碟子被搬動的步驟

「還有其他的碟子也有類似現象嗎？」

「3 號碟子隔八次出現。」

「所有的碟子有一致的規律嗎？」

「啊！我知道了！k 號碟子每隔 2^k 次搬動一次。」

「知道第 i 步驟搬動第幾號碟子嗎？」

「不知道，但是第 1，3，5，7，9，11，13，15 步是搬動第 1 號碟子，而第 2，6，10，14 步是搬動第 2 號碟子，第 4 和 12 步是搬動第 3 號碟子，而第 8 步搬動第 4 號碟子。」

「耶！我知道了！可能和這些數字的因數為 2 的倍數有關。」

「可以用符號寫下來嗎？」

「第 i 步驟搬動碟子號碼與 i 最大的因數 2^k 有關。」

我們終於知道，當進行第 i 步驟時，找出最大的 k 使得 $i=2^k\times z$ (z 為正整數)。則在第 i 步驟時，要搬動第 $k+1$ 號碟子。

「每次碟子搬動的方向，知道了嗎？」

「也許我們該看看，1 號碟子搬動方向的規律是什麼?」

圖 1.9 是 1 號碟子在整個搬動過程中的搬動方向。

步驟	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
移動的碟子	1	2	1	3	1	2	1	4	1	2	1	3	1	2	1
C			↑		↓				↑		↓				↑
B	↑		↑				↑		↑				↑		↑
A	↑				↓		↑				↓		↑		

圖 1.9 觀察 1 號碟子的搬動方向

「有規律嗎？」

「先看看 1 號碟子搬動的方向，好像往上移動到頂後就掉下來？」

「再看看 2 號碟子搬動的方向。有規律嗎？」

「1 號碟子和 2 號碟子搬動的方向有一些相像，但又不一樣。」

「3 號呢？」

「啊！我知道了！3 號碟子和 1 號碟子搬動的方式一致，也就是奇數號碟子搬動的方式一致。」

「偶數號碟子搬動的方式一致？」

「是的！偶數號碟子搬動的方式一致，只是方向正好相反。也就是往下移動到底後，就跳上去。」

圖 1.10 是偶數號碟子在整個搬動過程中的搬動方向。

步驟	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
移動的碟子	1	2	1	3	1	2	1	4	1	2	1	3	1	2	1
C		↑				↓		↑						↑	
B		↑				↓		↑		↓				↑	
A		↑						↑		↓				↑	

圖 1.10 觀察偶數號碟子的搬動方向

「試一試其他的例子，看看是否仍保留此規律呢？」

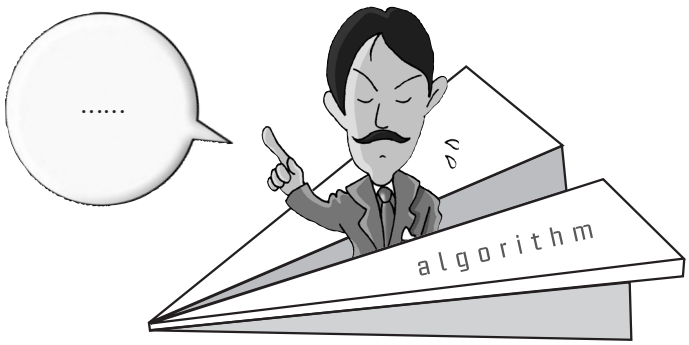
「當河內塔問題的碟子數是奇數時，好像整個搬動的方向全部相反。」

「可以換個方式描述您的觀察嗎？」

「當河內塔問題的碟子數是奇數時，奇數號碟子往下移動到底後，就跳上去；而偶數號碟子往上移動到頂後，就掉下來。」

「這些性質如何證明？」

「前面不是說，這本書不提及證明的嗎?」



根據以上的討論，可設計出一個解決河內塔問題的非遞迴演算法。為了方便起見，將三根柱子編號命名為 0(啟始的柱子)、1(可暫放的柱子)、2(目標的柱子)。

表 1.6 河內塔問題的非遞迴演算法

輸入	河內塔的碟子數 n 且碟子編號為{1，2，…，n}。三根柱子編號為 0，1，2
輸出	全部 2^n-1 次移動過程
步驟	一個迴圈控制整數 i 從 1 到 2^n-1 執行下列步驟來完成每一個碟子的移動： Step 1: 計算最大的 k 使得 $i=2^k \times y$。(故本次將搬動第 k+1 號碟子) Step 2: 當 n 為偶數且 k+1 為奇數時，移動第 k+1 號碟子自目前的 s 柱子搬到第 (s+1) MOD 3 柱子。 當 n 為偶數且 k+1 為偶數時，移動第 k+1 號碟子自目前的 s 柱子搬到第 (s-1) MOD 3 柱子。 當 n 為奇數且 k+1 為奇數時，移動第 k+1 號碟子自第 s 柱子搬到第(s-1) MOD 3 柱子。 當 n 為奇數且 k+1 為偶數時，移動第 k+1 號碟子自第 s 柱子搬到第(s+1) MOD 3 柱子。

1.4 發現演算法的技巧

我們只利用「觀察例子，找尋規律」就發現到解決河內塔問題的兩個演算法。你也許會問下列問題：

「我感覺好像比以前靈光多了？」

「你以前為什麼不靈光呢？」

「以前我只會努力記得別人教過的問題及方法。」

「現在不同了嗎？」

「有一點感覺，但是怎樣才有機會發現更多演算法呢？」

也許，坡利亞先生(G. Polya)在「如何解題」(How to Solve It) 中的一段話，可回答發現演算法的技巧。

如何解題 (How to Solve It)

(節錄及翻譯自坡利亞的「如何解題」一書)

首先，你必須了解問題

什麼是未知?什麼是資料?什麼是條件? 資料是否滿足這條件？條件是否足以決定未知？或者條件太冗贅或相互矛盾？繪一個圖。引用適當的符號來記錄。將條件分成好幾個部份。試著將它寫下來。

其次，找出已知和未知的關係。如果找不到立即的關係，你也許必須考慮輔助問題。最後你應該有一個解決的規劃。

你以前看過這個問題嗎? 或你看過相似的問題嗎？你看過相關問題嗎？你是否知道一個有用的定理。注視未知！嘗試想一個類似的問題有相同或相似的未知。這裡有一個相關而且知道解法的問題。你能利用它嗎？你能利用它的結果嗎？你能利用它的方法嗎？你可以引入一些輔助元素來方便利用它嗎？你可以重新表達這個問題嗎？你可以用更不同的方式表達這個問題嗎？

next

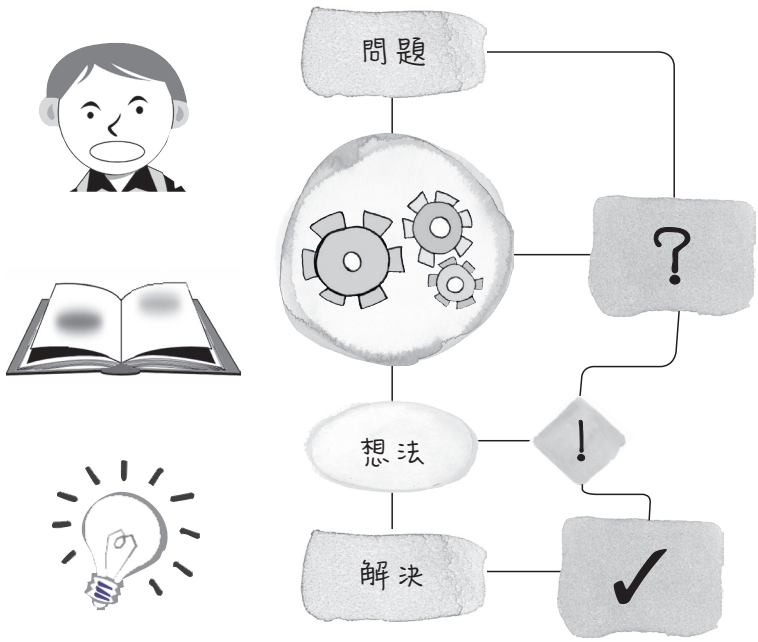
回歸到定義。如果你一時不能解決此問題，先嘗試一些相關的問題。你可以想像到一個更容易的相關問題嗎？一個更一般普遍的問題嗎？一個更特殊的問題嗎？一個類似的問題嗎？你可以解決問題的一部份嗎？留下一部份的條件而丟棄其他，距離未知有多遠呢？可以有多少變化呢？從已知可以推導出一些有用的東西嗎？你可以想像有其他資料合適於決定未知？你可以改變未知或已知(或同時改變)，如此新的未知和新的已知更接近對方？你是否利用了所有已知？你是否利用了所有條件？你是否考慮所有問題牽涉到的符號？

執行計劃

執行你解法的計劃，檢查每一個步驟。你可以看見每一個步驟是正確的嗎？你可以證明它的正確性嗎？

反省小心檢視所得的成果

你可以檢驗成果嗎？你可以檢驗爭議嗎？你可以將成果作不同的推導嗎？你可以將成果一眼看穿嗎？你可以將成果，或者導出這成果的方法，使用於其他的問題上嗎？



學習評量

1. 請撰寫(1)一個遞迴的程式，(2)和一個非遞迴的程式來解決河內塔問題，並比較其兩者所需之執行時間。請寫下您的觀察與想法。

2. 老王開雜貨店想送 N 個冬瓜糖磚給客戶。每個冬瓜糖磚長寬高都是10公分。老王希望將這些 N 個冬瓜糖磚包裝成一大包 ($x \times y \times z$ 的長方體)以方便運送，但為了愛台灣響應環保，希望使用的包裝紙愈少愈好。請寫一個程式輸入 N，輸出最少的包裝紙面積。

輸入：
9

輸出：
3000

3. 請寫一個程式將一正整數作質因數分解。例如，$3080=2^3 \times 5 \times 7 \times 11$。

輸入：
3080

輸出：
2^3 ×5×7×11