

河內塔是一個具三根杆子和幾個穿孔圓盤的東西，河內塔問題是：需把整個塔從原位移動到另一根杆子，規則是每次只能移動一個圓盤，而大的圓盤一定不能疊在小的上面，求共需要移動幾次。

算法是：假設有A、B、C塔，A塔有N塊圓盤，目標是把盤全部移動到C塔，所以需要先把A頂部的N-1塊圓盤移到B塔，再把A塔的最大塊圓盤移到C，再把B塔的N-1塊盤移到C塔，之後重複動作直到移完圓盤。

因為要將圓盤由下往上移動到目標位置，所以每當移完一個圓盤，就要繼續重複動作，直到每個圓盤都被移到目標塔，所以以C語言來看，他會在移動完一輪後，在程式末呼叫自己，以達到repeat的目的，因此以遞迴解決此問題是很好的做法。

維基百科的圖像解釋以無向圖來表示河內塔。每個節點表示盤子位置的一種可能性，每個邊表示一種移動方法。右圖為只有一個圓盤的河內塔的無相圖。兩個圓盤時，每個節點有兩個字母（代表兩個塔），數字由小到大代表圓盤大小，後面的字母表示較大的圓盤，且最初沒有被移動。如右圖：以這兩種情況，推論每多出一個圓盤，無向圖就會再多出兩個自己，形成一個新的三角形。

因為最長的路徑是 $2^n - 1$ 次，會浪費很多時間，因此無向圖很大的好處，就是能看出最短路徑。

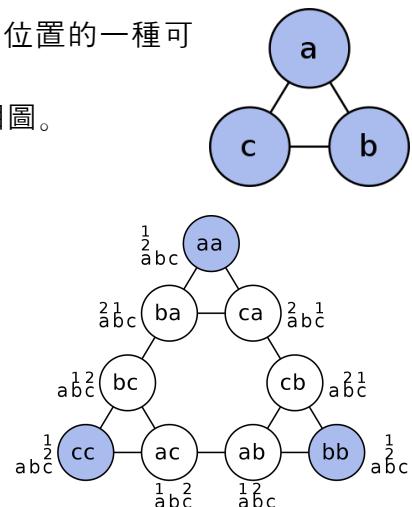
以數學方式來看：假設有三根柱子，其中一根柱子上有N個圓盤，共需移動T(N)次。

$$T(1)=1$$

$$T(2)=2T(1)+1=2+1=3$$

$$T(3)=2T(2)+1=2^2+2+1=7$$

$$T(4)=2T(3)+1=2^3+2^2+2+1=15$$



.....

$$T(N) = 2^{N-1} + 2^{N-2} + \dots + 2^3 + 2^2 + 2 + 1 = 2^N - 1$$

以「程式語言」方式來看，假設解n個圓盤的河內塔，執行次數為T(n)，首先要將上面的n-1個圓盤從A移動到B，為T(n-1)步，把最大的圓盤移動到C是一步，把位在B的n-1個圓盤移動到C是T(n-1)步，所以T(n)=2T(n-1)+1，可得T(n)= $2^n - 1$ 。

以下的結果是用AMD Ryzen 7 5800H跑的

---

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int moved=0;

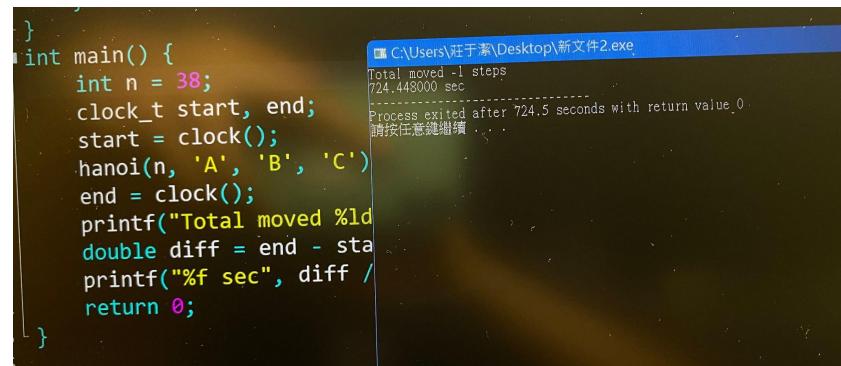
void hanoi(int n, char A, char B, char C) {
    if(n == 1) {
        moved++;
    }
    else {
        hanoi(n-1, A, C, B);
        hanoi(1, A, B, C);
        hanoi(n-1, B, A, C);
    }
}
```

```
int main() {  
  
    int n = 38;  
  
    clock_t start, end;  
  
    start = clock();  
  
    hanoi(n, 'A', 'B', 'C');  
  
    end = clock();  
  
    printf("Total moved %d steps\n", moved);  
  
    double diff = end - start;  
  
    printf("%f sec", diff / CLOCKS_PER_SEC);  
  
    return 0;  
  
}
```

---

Total moved -1 steps

724.448000 sec



```
int main() {  
    int n = 38;  
    clock_t start, end;  
    start = clock();  
    hanoi(n, 'A', 'B', 'C');  
    end = clock();  
    printf("Total moved %d steps\n", moved);  
    double diff = end - start;  
    printf("%f sec", diff / CLOCKS_PER_SEC);  
    return 0;  
}
```