

# 递归

xbZhong

[本页PDF](#)

## 汉诺塔问题

汉诺塔（又称河内塔）问题是源于印度一个古老传说的益智玩具。大梵天创造世界的时候做了三根金刚石柱子，在一根柱子上从下往上按照大小顺序摆着 64 片黄金圆盘。大梵天命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱子上。并且规定，在小圆盘上不能放大圆盘，在三根柱子之间一次只能移动一个圆盘。

现在路飞开始玩汉诺塔游戏，他放了  $n$  片黄金圆盘在第一根柱子上，从上到下依次编号为 1 到  $n$ ，1 号圆盘最小， $n$  号圆盘最大。路飞移动第  $i$  号圆盘的时候需要花费  $i$  点体力。现在路飞想把圆盘全部移动到第 2 根柱子上，移动过程中路飞必须遵守游戏规则。

现在路飞想知道他完成游戏的最小移动次数和最少消耗的体力。

在这道题中我们使用递归思想，因此只需要考虑边界条件以及  $n$  和  $n-1$  之间的关系，具体的实现过程我们不加考虑 > 我们知道，对于两个盘子的情况，我们可以直接移动。> 那么现在对于  $n$  个盘子的情况，我们将其看成最底下一个盘子和上面  $n-1$  个盘子两部分，因此我们可以将上面的  $n-1$  个盘子看成一个盘子 > \* 我们首先要做的就是将  $n-1$  个盘子移动到辅助塔，再将剩下的盘子移动到目标塔，再将  $n-1$  个盘子移动到目标塔 > \* 接着便是对剩下的  $n-1$  个盘子进行操作，将  $n-2$  个盘子移动到辅助塔，再将剩下的一个盘子移动到目标塔，再将  $n-2$  个盘子移动到目标塔 > \* 对于后面的情况，以此类推

因此 对于汉诺塔问题 可以将其分解成3步 \* 将  $n-1$  个盘子移动到辅助塔 \* 将剩下的一个盘子移动到目标塔 \* 将  $n-1$  个盘子移动到目标塔

可以写出代码

```
#include<iostream> using namespace std; int t=0; int p=0; int f(int n,char a,char b,char c) //将f (n) 看成n个盘子所需移动的次数 { if(n==1) {t++;p++;} else { f(n-1,a,c,b); //n-1个盘子移动到辅助塔 t++; //剩下的一个盘子移动到目标塔 p+=n; //消耗的体力 f(n-1,c,b,a); //n-1个盘子移动到目标塔 } } int main() { int n; char a,b,c; cin>>n; f(n,a,b,c); cout << t << " " << p; }
```

需要注意的是，这种代码的时间复杂度  $O(N)$  为  $O(2^n)$  代码运行效率太低 \*\*因此使用递推算法 可以知道  $f(n) = 2 * f(n-1) + 1$ ， $f(n)$  为  $n$  个盘子所需移动的次数  $g(n)=2^*g(n-1)+n$   $g(n)$  为移动  $n$  个盘子所需耗费的体力\*\*

```
#include<iostream> using namespace std; #define N 60 int a[N+1]; int b[N+1]; int main() { int n; a[1]=1; b[1]=1; cin >> n; for(int i=2;i<=n;i++) { a[i] =a[i-1]*2 + 1; b[i] =b[i-1]*2 + i; } cout<<a[n]<< " "<<b[n]; }
```

这个代码的时间复杂度为  $O(n)$ ，相对于前面的代码速度更快