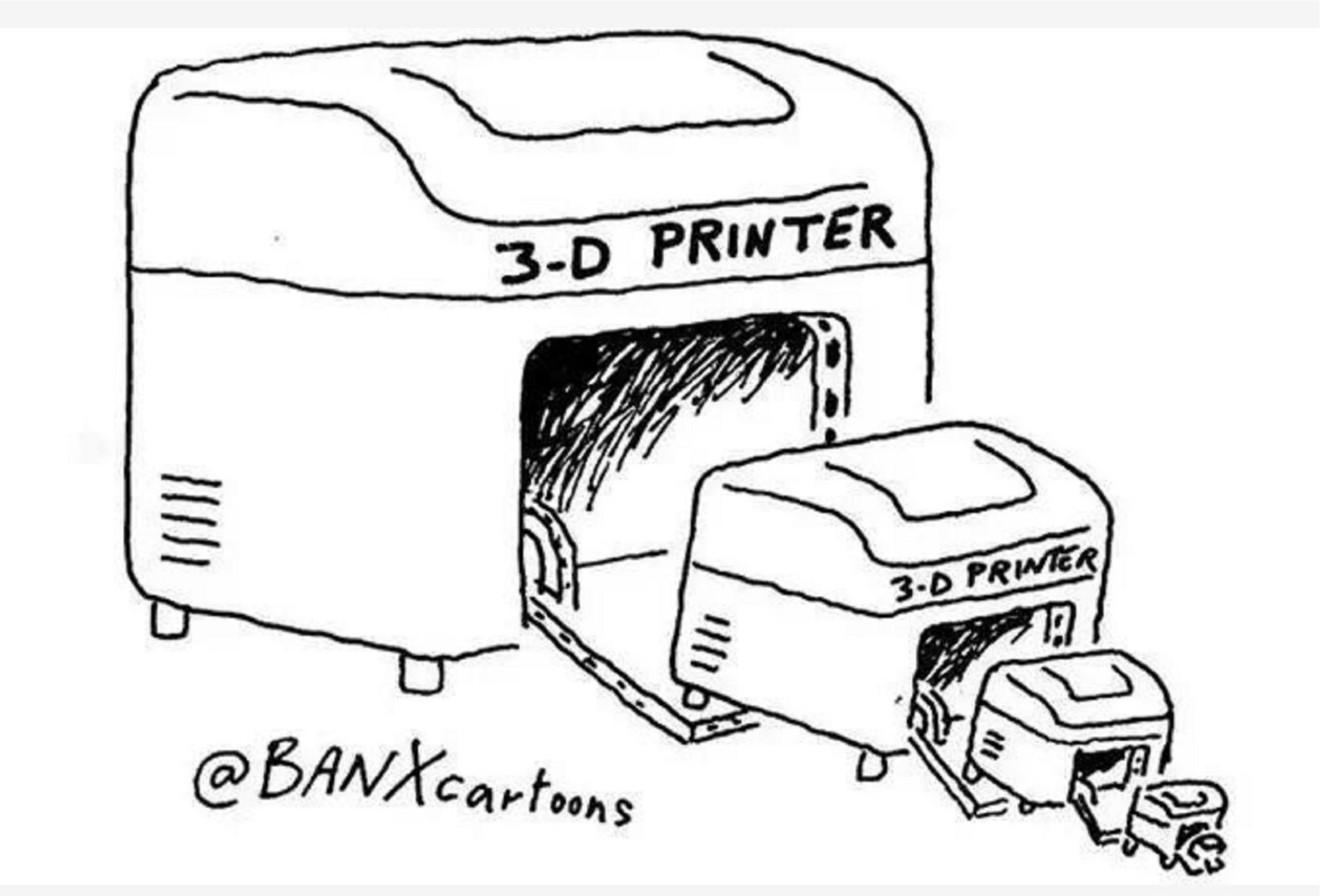
实验舱
青少年编程
走近科学 走进名校

挑战信息学奥林匹克

C++教程（304）
递归（1）

递归



递归-解决问题的方法

- 递归函数
  - 在函数中自己调用自己

- 递归算法
  - 用递归的思想解决问题

```
void fun1()
{
    ......;
    return;
}
int fun2()
{
    int x;
    fun1();
    ......;
    return x;
}
int main()
{
    int n;
    n = fun2();
    fun1();
    return 0;
}
```

认识递归

```cpp
void hs(int n)
{
    if(n < 4) hs(n+1);
    cout <<n << ": " << &n << endl;
}
int main()
{
    hs(1);
    return 0;
}
```

```
4:  0x6ffd70
3:  0x6ffda0
2:  0x6ffdd0
1:  0x6ffe00
```

认识递归

```
void hs(int n)
{
    cout <<n << ": " << &n << endl;
    if(n < 4) hs(n+1);
    cout <<n << ": " << &n << endl;
}
int main()
{
    hs(1);
    return 0;
}
```

```
1: 0x6ffe00
2: 0x6ffdd0
3: 0x6ffda0
4: 0x6ffd70
4: 0x6ffd70
3: 0x6ffda0
2: 0x6ffdd0
1: 0x6ffe00
```

写出阶乘问题的递归函数

n! = 1*2*3*4……*(n-1)*n

函数：

f(1) = 1

f(2) = 1*2

f(3) = 1*2*3

…………

f(n) = 1*2*3*4……*(n-1)*n

$$f(n)=\begin{cases} 1 & (n = 1) \\ n*f(n-1) & (n > 1) \end{cases}$$

写出阶乘问题的递归函数

递归边界条件

```
long long jc(int n)
{
    if ( n == 1 ) return 1;
    return n * jc( n - 1 );
}
```

递归调用

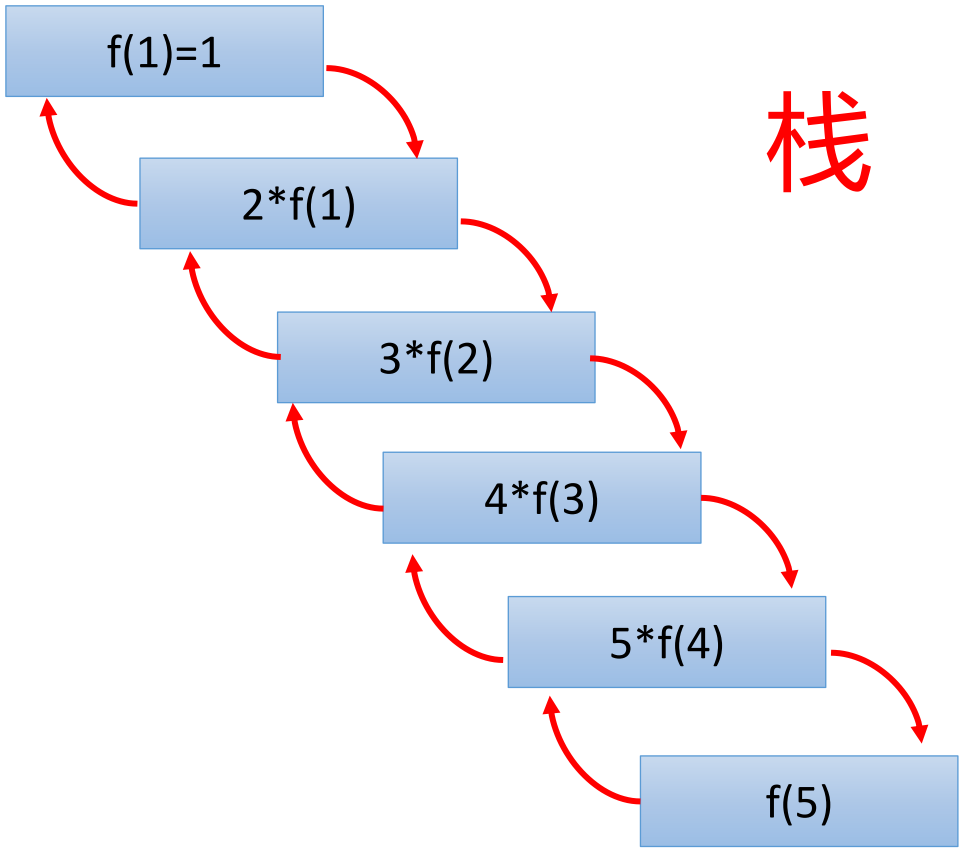
递归函数的条件

- 存在递归关系，问题可以划归为一个子问题。
- 递归有终止的条件（边界条件）。

```cpp
#include <bits/stdc++.h>
using namespace std;

long long jc(int n)
{
    if ( n == 1 ) return 1;
    return n * jc( n - 1 );
}

int main()
{
    int n;
    cin >> n;
    cout << jc(n) << endl;
    return 0;
}
```

阶乘问题

```cpp
long long jc(int n)
{
    if ( n == 1 ) return 1;
    return n * jc( n - 1 );
}

int main()
{
    int n;
    cin >> n;
    cout << jc(n) << endl;
    return 0;
}
```



栈

Fibonacci数列（Leonardo Fibonacci）

0，1，1，2，3，5，8，13，21，34， 55……

- 递归关系

fib(n) = fib(n - 1) + fib(n - 2)   n >= 2

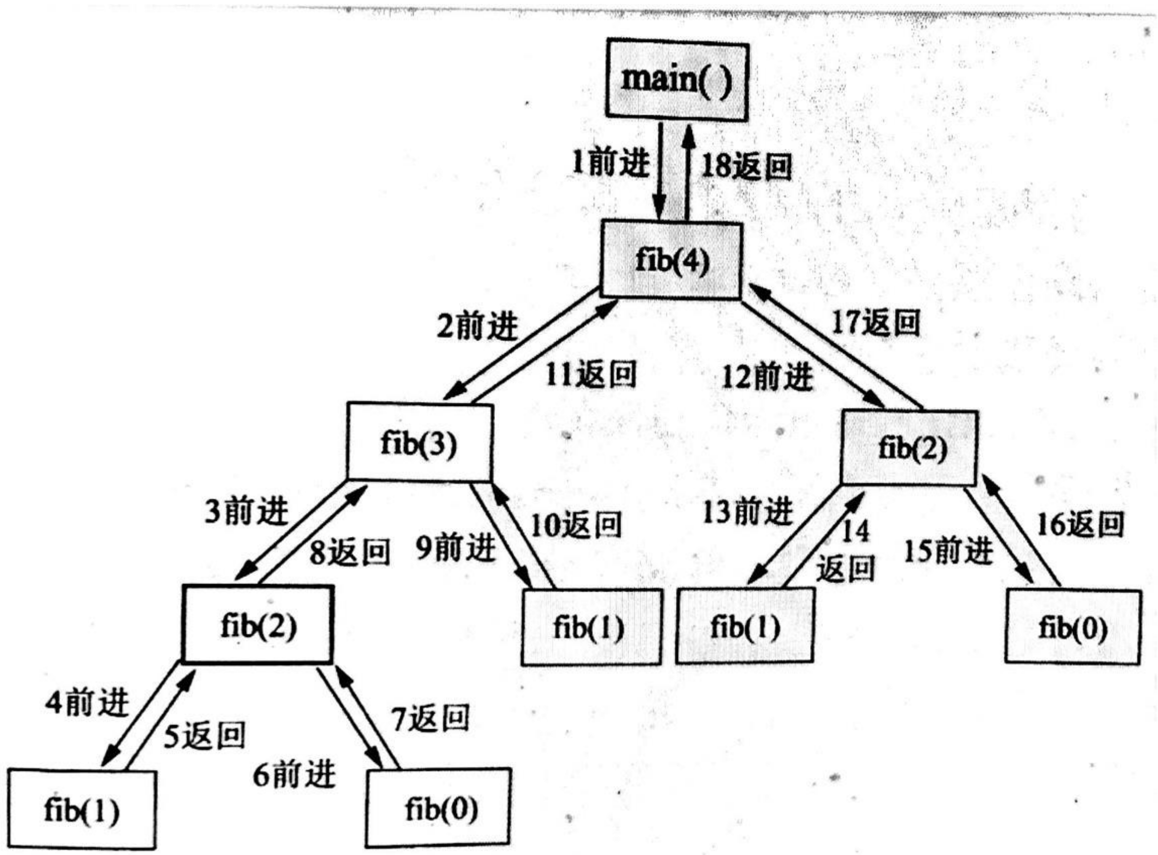
- 边界条件

fib(0) = 0   n = 0

fib(1) = 1   n = 1

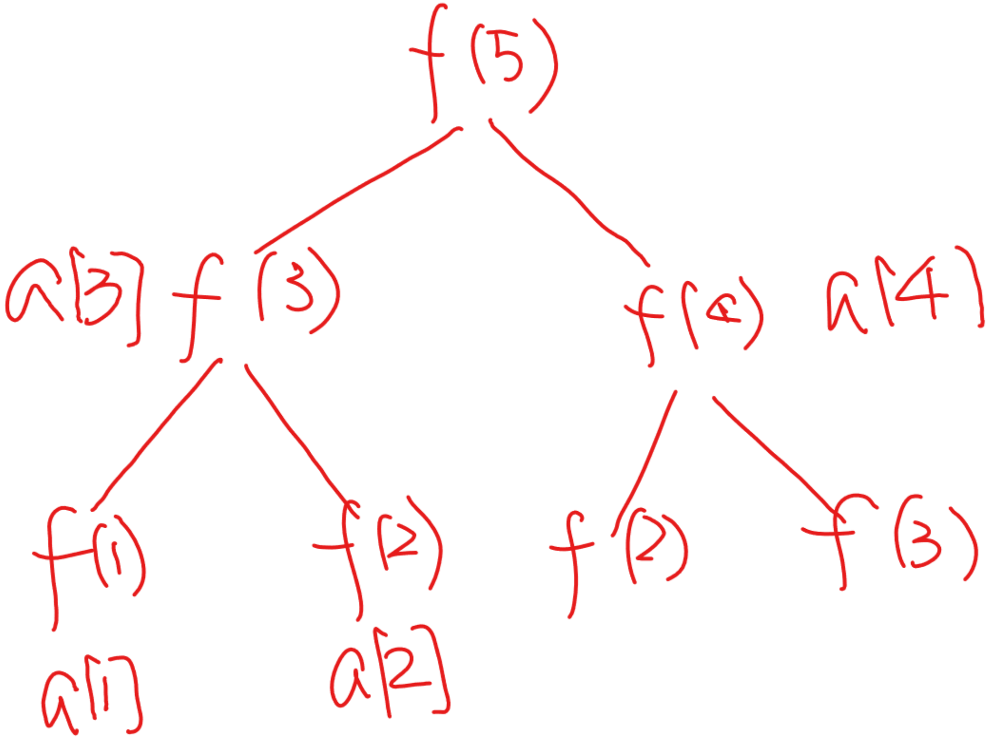
Fibonacci数列

```
int fb(int t)
{
    if (t==0) return 0;
    if (t==1) return 1;
    return fb(t-2)+fb(t-1);
}
int main()
{
    int n;
    cin >>n;
    cout <<fb(n)<<endl;
    return 0;
}
```



记忆化递归

```
long long a[100];
long long fb(int n)
{
    if (a[n]>0) return a[n];
    a[n]=fb(n-2)+fb(n-1);
    return a[n];
}
int main()
{
    int m;
    cin >>m;
    a[1]=a[2]=1;
    cout <<fb(m)<<endl;
    return 0;
}
```



例1：爬楼梯问题

- 楼梯有n级台阶
- 可以一步跨一级台阶，也可以一步跨二级台阶
- 问：n级台阶有多少种走法
- 由于最后答案可能很大，输出最后的答案对 100007 取模的结果
- $1 <= n <= 1000$

参考代码

```
const int k = 100007;
```

```
long long a[100];
long long fb(int n)
{
    if (a[n]>0) return a[n];
    a[n]=fb(n-2)+fb(n-1);
    return a[n];
}
int main()
{
    int m;
    cin >>m;
    a[1]=a[2]=1;
    cout <<fb(m)<<endl;
    return 0;
}
```

```
(fb(n-1) % k + fb(n-2) % k ) % k;
```

最大公约数

```
int Gcd(int x,int y)
{
    if (y == 0) return x;
    return Gcd(y, x % y);
}
```

用递归代替循环

```
int gcd(int x,int y)
{
    int r=x%y;
    while (r!=0)
    {
        x=y;
        y=r;
        r=x%y;
    }
    return y;
}
```

例2：最大公约数和最小公倍数问题

描述

输入二个正整数x0, y0(2 <= x0 < 100000, 2 <= y0 <= 1000000), 求出满足下列条件的p, q的个数条件:

1. P, Q是正整数

2. 要求P, Q以x0为最大公约数, 以y0为最小公倍数。

试求: 满足条件的所有可能的两个正整数的个数。

输入

一行，包含两个正整数x0和y0，中间用单个空格隔开。

输出

一个整数，即满足条件的个数。

样例输入
3 60
样例输出
4
提示
此时的P Q分别为:
3 60
15 12
12 15
60 3
所以:满足条件的所有可能的两个正整数的个数共4种。

问题分析

1. 原数n = x0*y0
2. 枚举n所有的因子对(x, y)，验证x, y的公约数是否等于x0，并计数。
3. 这个算法会超时，需优化算法。
4. 观察样例，所有的P，Q数对除以公约数x0后，都是互质的，所有取 n = y0 / x0
5. 枚举n所有的因子对(x, y)，验证x, y是否互质
6. 考虑到程序的时间效率，枚举因子时最大数取sqrt(n)
7. 输出时乘以2即可
8. 考虑特殊情况：y0%x0!=0, P,Q不存在

样例输入
3 60
样例输出
4
提示
此时的P Q分别为:
3 60
15 12
12 15
60 3
所以:满足条件的所有可能的两个正整数的个数共4种。

算法优化

```
int x0, y0, n, m, s = 0;
cin >> x0 >> y0;
if ( y0 % x0 != 0)
{
    cout << 0 << endl;
    return 0;
}
n = y0 / x0;
m = sqrt(n);
for ( int i = 1; i <= m; i++ )
{
    if ( n % i == 0 && Gcd(i, n/i) == 1) s++;
}
cout << s * 2 << endl;
```

例2：第k个数

【描述】

输入一个正整数n，输出从右边数第k个数字。

【输入】

2个整数，分别是n和k。(n<10^9)

【输出】

一个一位数。

【输入样例】

394829 3

【输出样例】

8

问题分析

- 分离数字n
  - n % 10
  - n / 10

```
int shu(int t, int k)
{
    if (k == 1) return t % 10;
    return shu(t / 10, k - 1);
}
```

第k个数

```
int shu(int t, int k)
{
    if (k == 1) return t % 10;
    return shu(t / 10, k - 1);
}
int main()
{
    int n, m;
    cin >> n >> m;
    cout << shu(n, m) << endl;
    return 0;
}
```

举一反三：递归转二进制

```
void Binary(long long x)
{
    if (x == 0) return;
    else Binary(x / 2);
    cout << x % 2;
}
```

替代循环
不用数组

例3：放苹果

描述

把M个同样的苹果放在N个同样的盘子里，允许有的盘子空着不放，问共有多少种不同的分法？（用K表示）5，1，1和1，5，1 是同一种分法。

输入

第一行是测试数据的数目t（0 <= t <= 20）。以下每行均包含二个整数M和N，以空格分开。1<=M，N<=10。

输出

对输入的每组数据M和N，用一行输出相应的K。

样例输入

1

7 3

样例输出

8

问题分析

- 设f(m,n) 为m个苹果，n个盘子的放法数目；
- 当n>m：
  - 必定有n-m个盘子永远空着，去掉它们对摆放苹果方法数目不产生影响。f(m,n) = f(m,m)
- 当n<=m：
  - 不同的放法可以分成两类：
    1. 有空盘子，即相当于f(m,n-1);
    2. 所有盘子都有苹果，相当于每个盘子中先放一个苹果，问题归结为m-n个苹果放n个盘子，即f(m-n,n)；

  f(m,n) =f(m,n-1)+f(m-n,n)
- 边界条件：
  - 当n=0时，没有盘子可放，所以返回0；
  - 当m=0，没有苹果可放，定义为 1 种放法；

递归函数

```c
int f(int m,int n)
{
    if( n > m ) return f(m,m);
    if( m == 0) return 1;
    if( n <= 0 ) return 0;
    return f(m,n-1) + f(m-n,n);
}
```

汉诺塔游戏

汉诺塔由编号为1到n大小不同的圆盘和三根柱子a,b,c组成，编号越小盘子越小。开始时，这n个圆盘由大到小依次套在a柱上，如图1.6.3所示。要求把a柱上n个圆盘按下述规则移到c柱上：

①一次只能移一个圆盘，它必须位于某个柱子的顶部；

②圆盘只能在三个柱子上存放；

③任何时刻不允许大盘压小盘。

将这n个盘子用最少移动次数从a柱移动到c柱上，输出每一步的移动方法。

输入：只有一行，一个整数n(1<=n<=20),表示盘子的数量。

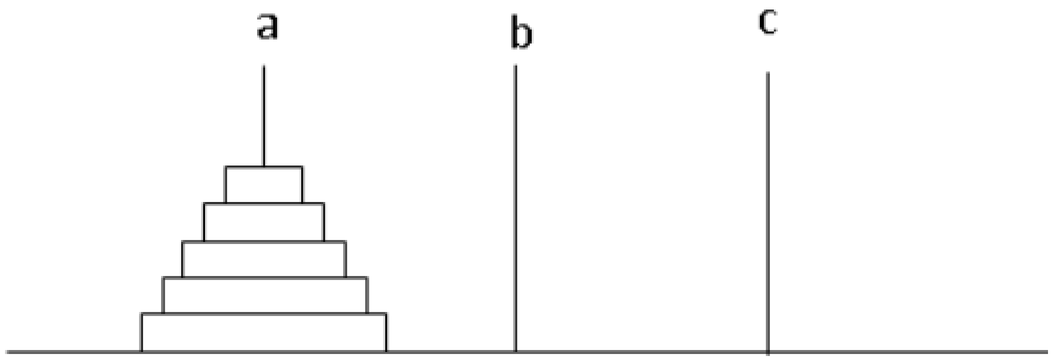
输出：输出若干行，每一行的格式是“步数.Move

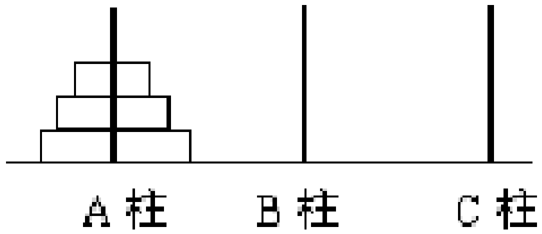
样例输入

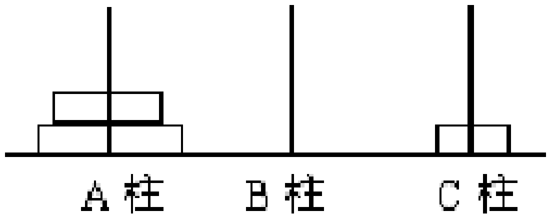
3

样例输出

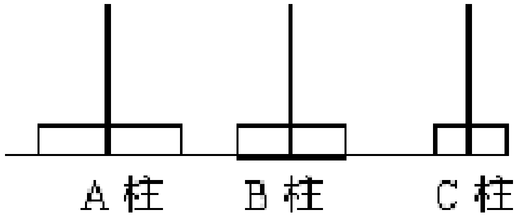
```
1. Move 1 from A to C
2. Move 2 from A to B
3. Move 1 from C to B
4. Move 3 from A to C
5. Move 1 from B to A
6. Move 2 from B to C
7. Move 1 from A to C
```



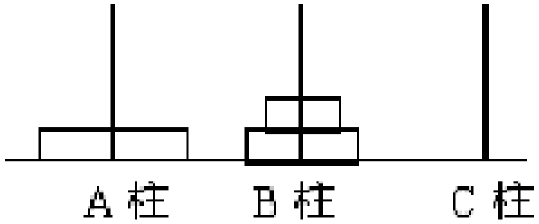




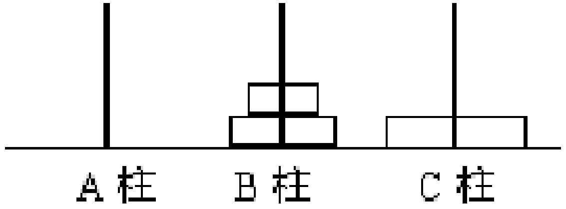
第 1 步： A→C



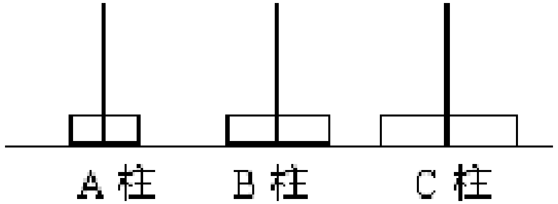
第 2 步： A→B



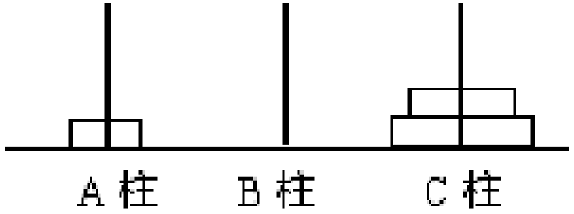
第 3 步： C→B



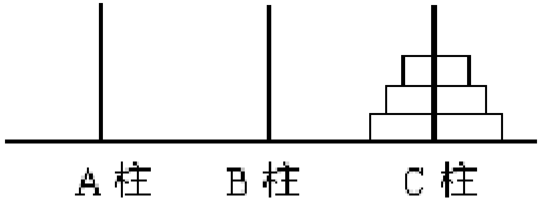
第 4 步： A→C



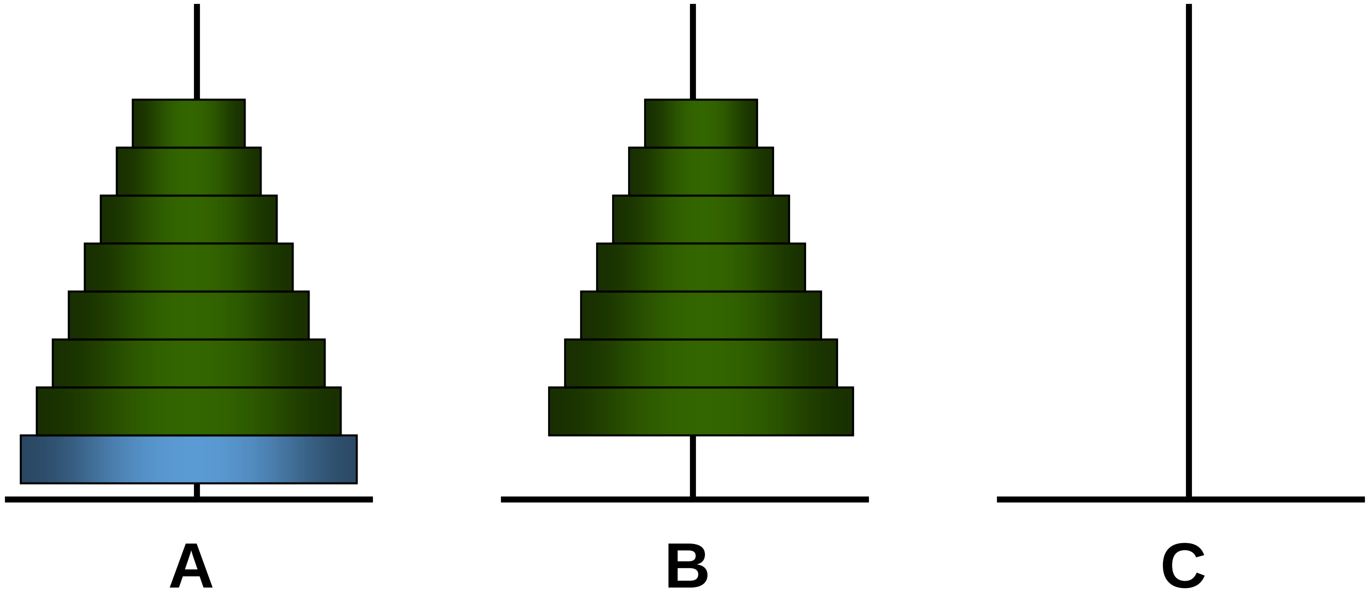
第 5 步： B→A



第 6 步： B→C



第 7 步： A→C



1: (n-1, A, C, B)

3: (n-1, B, A, C)

2: A $\rightarrow$ C

汉诺塔

```cpp
void hanoi(int n, char a, char b, char c)
{
    if ( n == 0 ) return;
    else
    {
        hanoi(n-1, a, c, b);
        cout << ++step << ':' << a << "->" << n << "->" << c << endl;
        hanoi(n-1, b, a, c);
    }
}
```