



程序设计实习

郭炜 微博 <http://weibo.com/guoweiofpku>

<http://blog.sina.com.cn/u/3266490431>

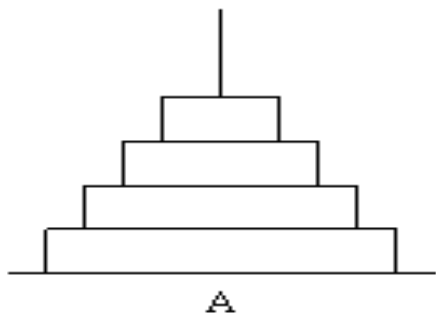
刘家瑛 微博 <http://weibo.com/pkuliujiaying>



用栈替代递归

汉诺塔问题

古代有一个梵塔，塔内有三个座A、B、C，A座上有64个盘子，盘子大小不等，大的在下，小的在上（如图）。有一个和尚想把这64个盘子从A座移到B座，但每次只能允许移动一个盘子，并且在移动过程中，3个座上的盘子始终保持大盘在下，小盘在上。在移动过程中可以利用B座，要求输出移动的步骤。



汉诺塔问题递归解法

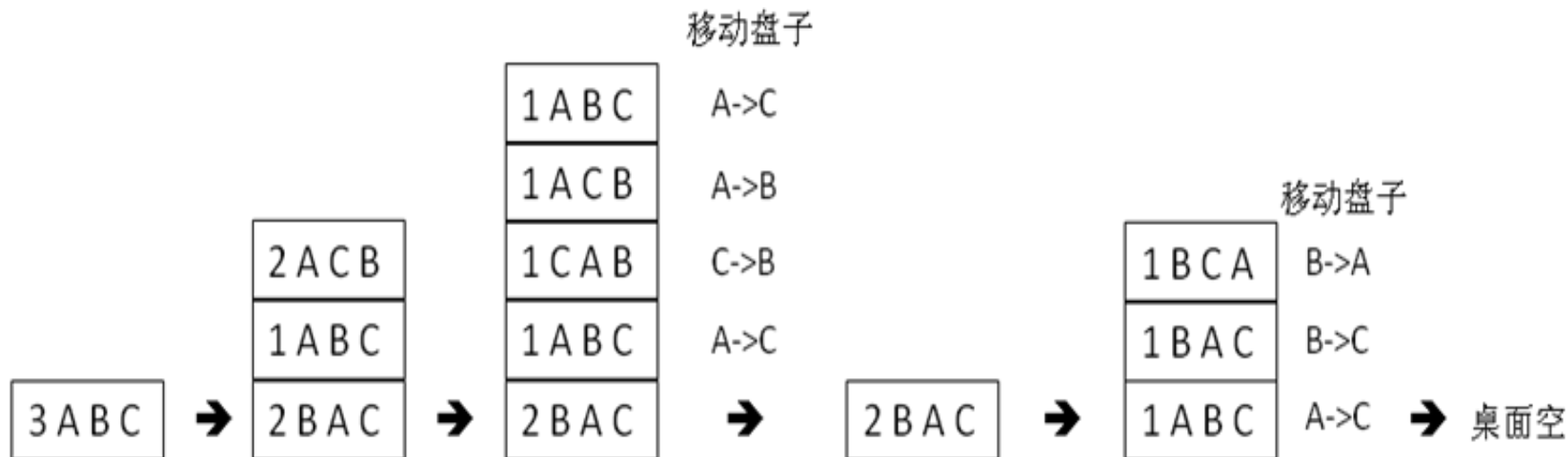
```
#include <iostream>
using namespace std;
void Hanoi(int n, char src, char mid, char dest)
//将src座上的n个盘子，以mid座为中转，移动到dest座
{
    if( n == 1) { //只需移动一个盘子
        cout << src << "->" << dest << endl; //直接将盘子从src移动到dest即可
        return ; //递归终止
    }
    Hanoi(n-1, src, dest, mid); //先将n-1个盘子从src移动到mid
    cout << src << "->" << dest << endl; //再将一个盘子从src移动到dest
    Hanoi(n-1, mid, src, dest); //最后将n-1个盘子从mid移动到dest
    return ;
}
```

汉诺塔问题递归解法

```
int main()
{
    int n;
    cin >> n; //输入盘子数目
    Hanoi(n,'A','B','C');
    return 0;
}
```

汉诺塔问题手工解法(三个盘子)

信封堆,每个信封放一个待解决的问题



汉诺塔问题非递归解法

```
#include <iostream>
#include <stack>
using namespace std;
struct Problem {
    int n;
    char src,mid,dest;
    Problem( int nn, char s,char m,char d ):n(nn),src(s),mid(m),dest(d) { }
}; //一个Problem变量代表一个子问题，将src上的n个盘子，
    // 以mid为中介，移动到dest
stack<Problem> stk;           //用来模拟信封堆的栈，一个元素代表一个信封
                             //若有n个盘子，则栈的高度不超过  $n * 3$ 
```

```

int main() {
    int n;  cin >> n;
    stk.push(Problem(n,'A','B','C')); //初始化了第一个信封
    while( ! stk.empty()) { //只要还有信封，就继续处理
        Problem curPrb = stk.top(); //取最上面的信封，即当前问题
        stk.pop(); // 丢弃最上面的信封
        if( curPrb.n == 1 )  cout << curPrb.src << "->" << curPrb.dest << endl ;
        else { //分解子问题
            //先把分解得到的第3个子问题放入栈中
            stk.push(Problem(curPrb.n -1,curPrb.mid,curPrb.src,curPrb.dest));
            //再把第2个子问题放入栈中
            stk.push(Problem(1,curPrb.src,curPrb.mid,curPrb.dest));
            //最后放第1个子问题，后放入栈的子问题先被处理
            stk.push(Problem(curPrb.n -1,curPrb.src,curPrb.dest,curPrb.mid));
        }
    }
    return 0;
}

```


汉诺塔问题递归解法

编译器生成的代码自动维护一个问题的栈，相当于信封堆。栈里每个子问题的描述中多了一项 --- 返回地址，返回地址可以描述该子问题已经解决到哪个步骤了，下面的(0) (1),(2),(3)就是返回地址

```
void Hanoi(int n, char src,char mid,char dest)
{
    if( n == 1) {
        (0)cout << src << "->" << dest << endl;
        return ;
    }
    Hanoi(n-1,src,dest,mid);
    (2)cout << src << "->" << dest << endl;
    Hanoi(n-1,mid,src,dest);
    (3)return ;
}
```

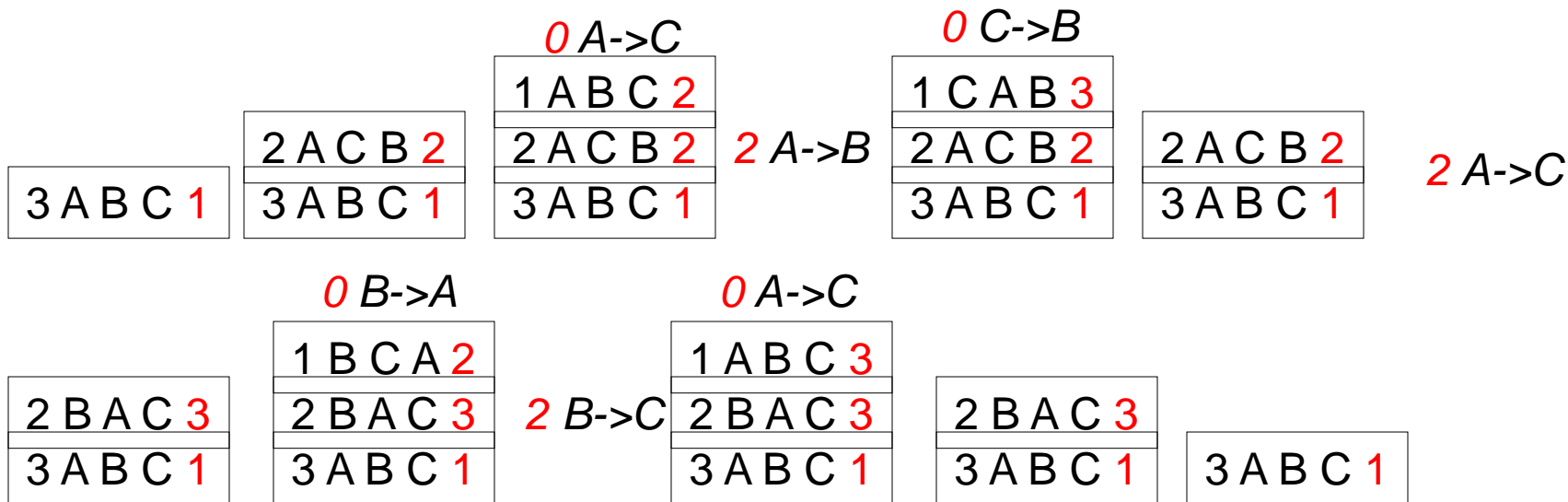
```
int main() {
    int n;
    cin >> n;
    Hanoi(n,'A','B','C');
    (1) return 0;
}
```

main中调用Hanoi时，栈形成初始状态：

n	A	B	C	1
---	---	---	---	---

汉诺塔问题递归解法

$n = 3$ 时，栈的变化状态:



```
void Hanoi(int n, char src, char mid, char dest) {  
    if( n == 1) { (0)cout << src << "->" << dest << endl;    return ;    }  
    Hanoi(n-1,src,dest,mid);  
    (2)cout << src << "->" << dest << endl;  
    Hanoi(n-1,mid,src,dest);  
    (3)return ;  
}
```

```
int main() {  
    int n;  
    cin >> n;    Hanoi(n,'A','B','C');  
    (1) return 0;  
}
```