1205：汉诺塔问题

题目描述

约19世纪末，在欧州的商店中出售一种智力玩具，在一块铜板上有三根杆，最左边的杆上自上而下、由小到大顺序串着由64个圆盘构成的塔。目的是将最左边杆上的盘全部移到中间的杆上，条件是一次只能移动一个盘，且不允许大盘放在小盘的上面。

这是一个著名的问题，几乎所有的教材上都有这个问题。由于条件是一次只能移动一个盘，且不允许大盘放在小盘上面，所以64个盘的移动次数是：18,446,744,073,709,551,615

这是一个天文数字，若每一微秒可能计算(并不输出)一次移动，那么也需要几乎一百万年。我们仅能找出问题的解决方法并解决较小N值时的汉诺塔，但很难用计算机解决64层的汉诺塔

假定圆盘从小到大编号为1，2，...

输入

输入为一个整数(小于20）后面跟三个单字符字符串。

整数为盘子的数目，后三个字符表示三个杆子的编号。

输出

输出每一步移动盘子的记录。一次移动一行。

每次移动的记录为例如 a->3->b 的形式，即把编号为3的盘子从a杆移至b杆。

输入样例

2 a b c

输出样例

a->1->c
a->2->b
c->1->b

分析

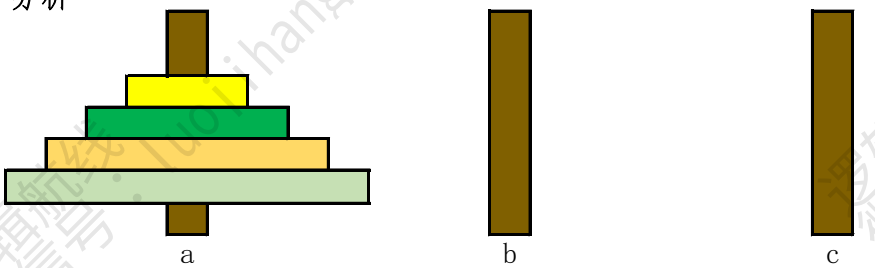


图1

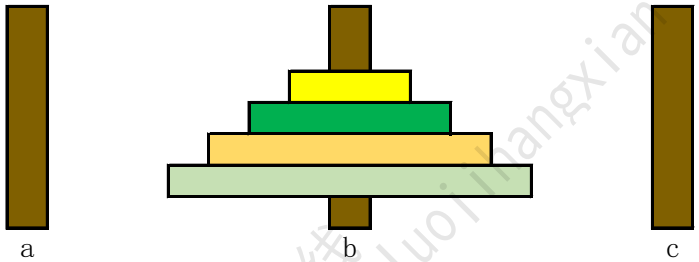


图2

汉诺塔要求的就是从图1的状态到达图2的状态，总共需要操作多少步？其实，想要彻底搞懂汉诺塔，你就必须要搞懂递归思维，那么如何来理解递归思维呢？你需要有分工思想！

假设，你的老板给了你一堆工作，如果完全你自己来做很可能费时又费力，但是如果你能够把它分配出去，那么就会极大的提高效率。理解递归思想最重要的一部分就是如何分配你手里的工作。

现在，我们就以汉诺塔为例，来理解递归中的工作分配。

首先，老板给你的工作是把总数为n圆盘从a柱经过c柱移动到b柱。对于你来说，整个工作流中最简单的任务就是**只搬最下面的一块。**那么，怎么样才能做到让自己只搬最下面的一块呢？

答案非常的简单：给自己留下最下面的那个圆盘，其余的n-1块圆盘交给别人处理，你无需理会！

想象一下这个场景。你在办公室叫来你的组员：“小王，你把这n-1块圆盘搬走！嗯，就放到c上吧。别耽误我放第n块圆盘到b上。”

小王挠了挠头：“这么多我也搬不完啊？”

你机智的笑了笑：“你手底下不是还有个实习生吗？让他和你一块弄！”

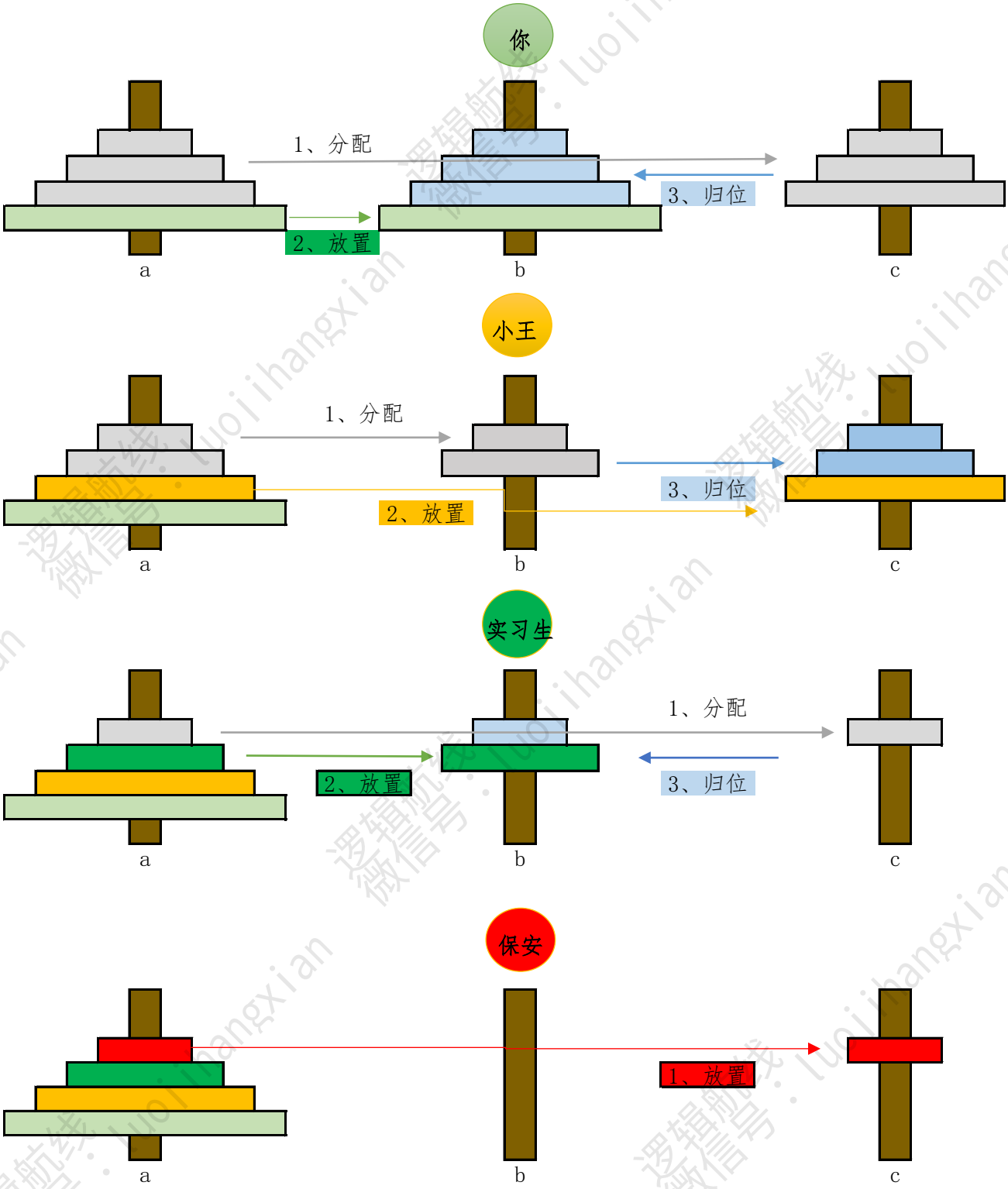
小王立马如梦初醒：“领导，我明白了，这就去办！”

就这样，你找了小王，小王找了实习生，实习生又找了保安大爷，保安大爷又拽上了大黄……在你们一群人的努力下，终于完成了工作！

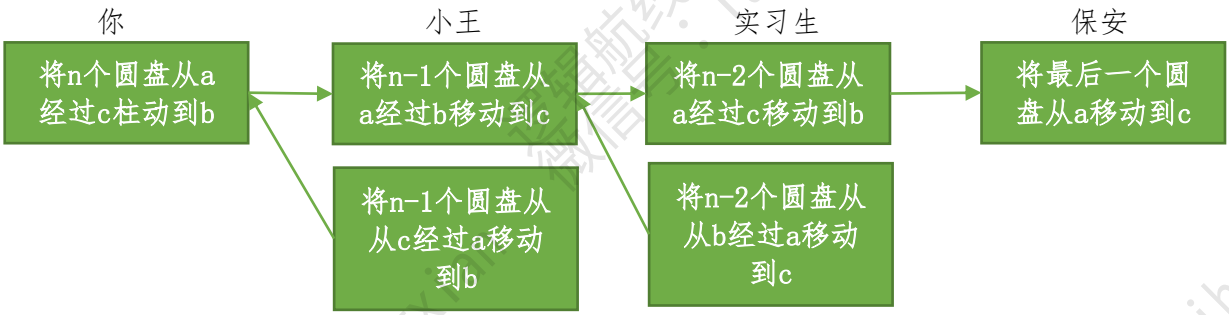
在这里，每个都只挪动自己任务中的最后一块，而其余的则交给别人，每个人又都有明确的起点位置、终点位置以及经停位置，他们的工作可以抽象成如下的函数：

```
移动汉诺塔(数量,起点为止，终点位置，经停位置)
{
    步骤1   助手.移动汉诺塔(数量-1,起点位置，经停位置，终点位置)
    步骤2   你将最后一片放置在终点位置
    步骤3   助手.移动汉诺塔(数量-1,经停位置，终点位置，起点位置)
}
```

我们再用一张图来诠释以下这个过程：



在这里每个人都只关注自己目标中最下方的一块，而把上面的分配出去，待上面的工作完成后，自己才能完成工作，当自己完成后，再将上面的盘子归位。于是我们不断的分配、归位，最后就能够将汉诺塔移动完成.下面是文字流程描述。



上面的伪代码诠释了移动汉诺塔的基本操作。这里解释一下助手小王在执行移动汉诺塔时，第三个参数为什么是经停位置？同时也是上图中不同角色来回切换目标点的原因。

我们可以看到函数的原始定义，它的第三个参数的本意是终点位置，即我当前要把盘子安放的那个柱子位置。由于起点柱子和目标柱子都已经被我锁定了，所以助手小王不能把盘子放到这两根柱子上，因此他别无选择，只能把他手中的盘子放到我的经停柱子上。

对于实习生也是如此，他不能占用小王的起点和终点，因此实习生的终点就变成了小王的经停位置。以此类推，每个人的终点都是上一个人的经停位置。

最后一个问题：什么时候分配停止？很容易想到，那就是接到任务的人它的任务只有一个盘子，那么它只要直接把盘子放到目标柱子上就可以了，例如上图中的保安！

我们再把伪代码完善一下：

```
移动汉诺塔(数量,起点为止，终点位置，经停位置)
{
    如果(数量==1)
    {
            你将最后一片放置在终点位置
            退出任务
    }
   步骤1   助手.移动汉诺塔(数量-1,起点位置，经停位置，终点位置)
   步骤2   你将最后一片放置在终点位置
   步骤3   助手.移动汉诺塔(数量-1,经停位置，终点位置，起点位置)
}
```

编码

```cpp
#include<bits/stdc++.h>

using namespace std;

//打印当前的移动信息
void move(int n, char start, char end) {
    if (n <= 0)
        return;
    printf("%c->%d->%c\n", start, n, end);
}

//汉诺塔递归函数
//将n片从start点经停temp点到达end点
void hanoi(int n, char start, char end, char temp) {
    if (n > 0) //递归结束标
    {
        hanoi(n - 1, start, temp, end);  //将任务分配给助手
        move(n, start, end);   //自己完成最后一步
        hanoi(n - 1, temp, end, start);   //助手再将其余的盘片放置于上方
    }
}

int main() {
    int n;
    char a, b, c;
    cin >> n;
    cin >> a >> b >> c;
    hanoi(n, a, b, c);
    return 0;
}
```