从易到难的分析策略与汉诺塔问题

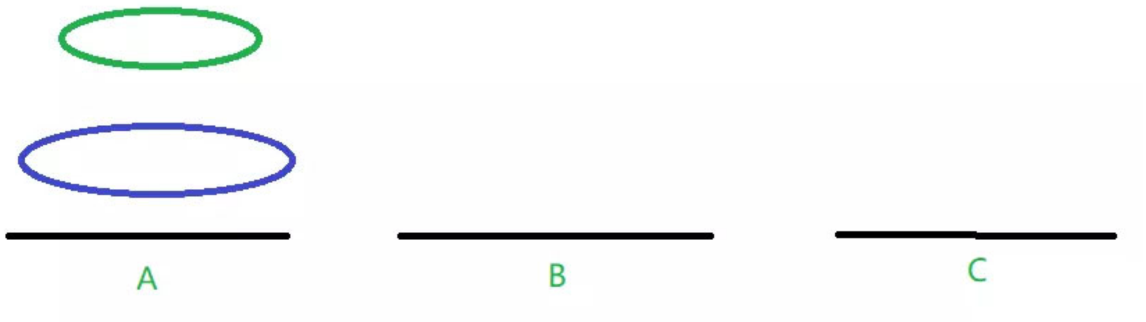
原创 乔祥硕 LOA算法学习笔记 2021-12-09 23:46

现有这样一个问题：上帝创造了三根柱子，并在第一根柱子上按顺序套有N个大小不同的圆盘（自下而上，圆盘由大到小，呈金字形）。规定每次只能移动最顶端的一个圆盘，并且保证整个过程中大圆盘不能放在小圆盘之上。欲将所有圆盘从第一根柱子移动到第三根柱子，试给出解决方案。将三个柱子分别记为A，B，C，原问题可简化为：

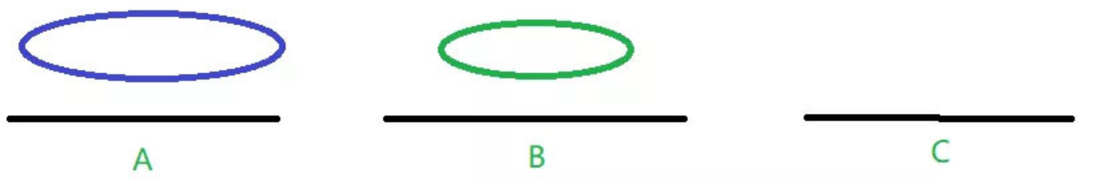
> 将N个大小不同的圆盘由A移动到C，每次只允许移动最顶端的单个圆盘，同一柱子上不得打乱圆盘的原有上下位置。

我们首先来进行如下的分析（多图预警）：我们从最简单的情况开始分析。如果只有一个圆盘，那么操作很简单，只需要直接把圆盘自A移动到C，也就是$A \to C$。

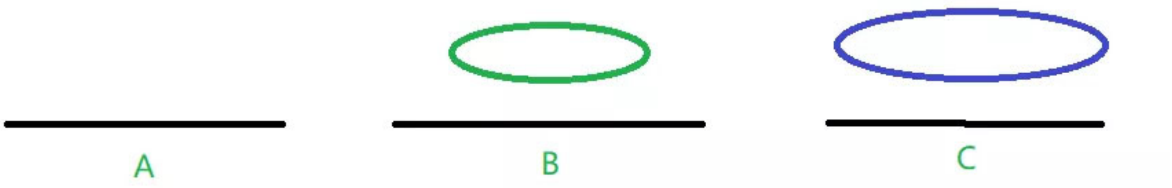
如果有两个圆盘，那么大概就是这个样子：



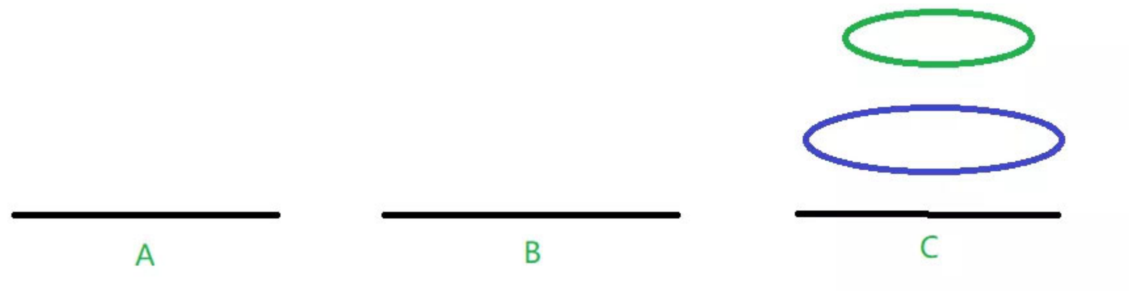
这时候，我们首先把绿圆盘从A移动到B：



再把紫色圆盘从A移动到C：

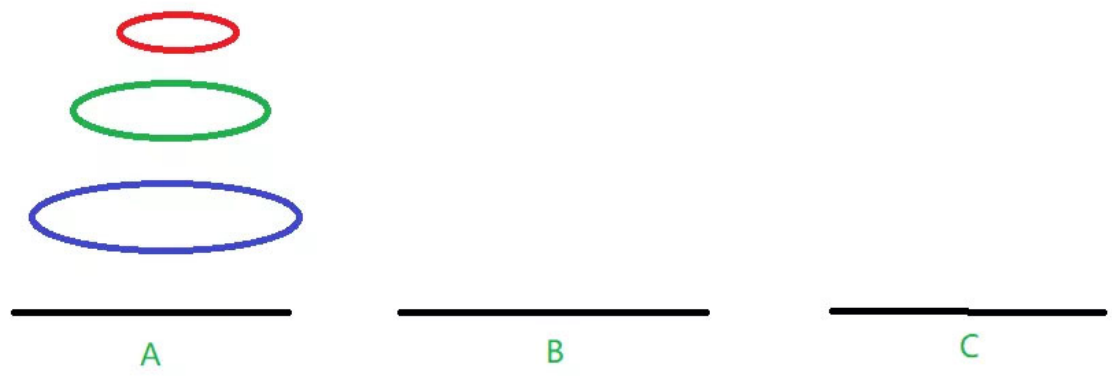


最后再把绿色圆盘移动到C即可：

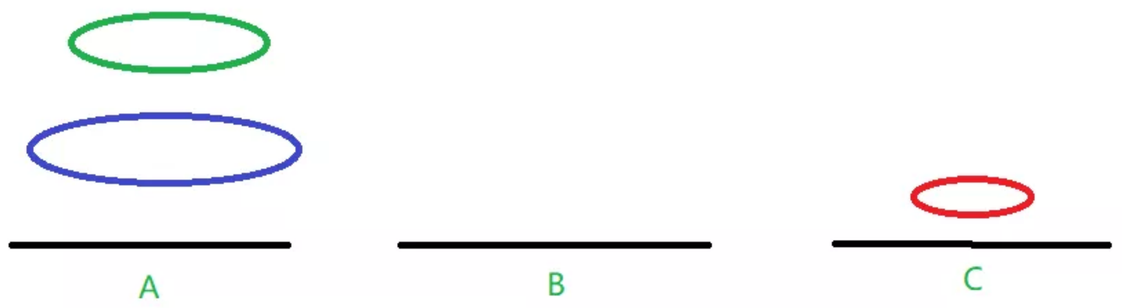


就这样，两个圆盘的问题，我们已经解决了。

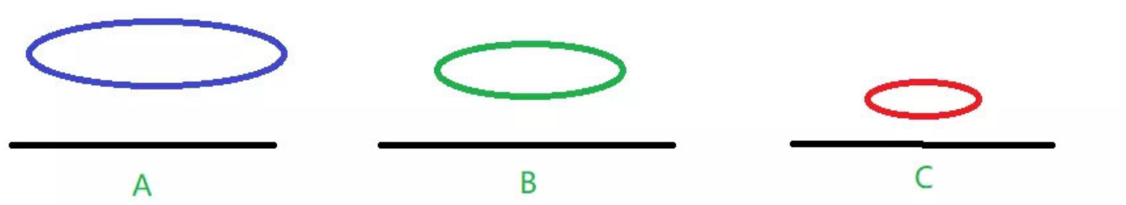
接下来，我们来研究三个圆盘的情景。



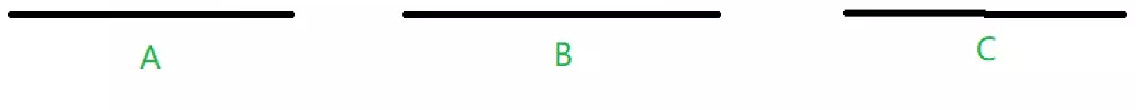
把红色圆盘移动到C：



再把绿色圆盘移动到B：

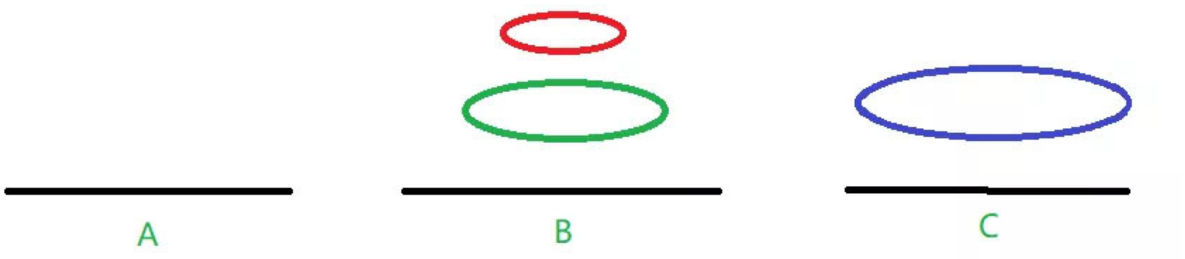


再把红色圆盘从C移动到B：



在这里我们先停一下。由之前的几步操作，我们实际上是把A最上面的两个圆盘从A移动到了B。

我们继续移动，把最大的紫色圆盘移动到C：



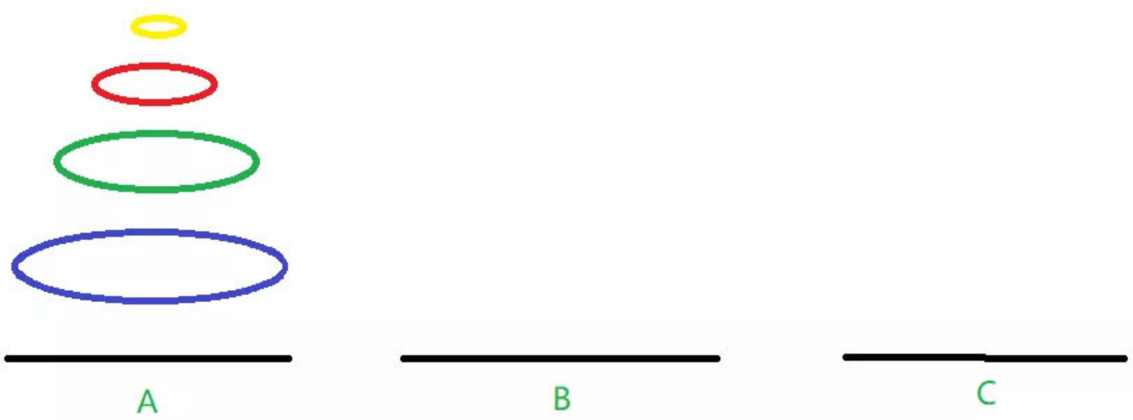
我们再停一下，分析一下接下来我们该怎样移动。

从图中我们可以看出，只需要再把B上的两个圆盘移动到C就可以了，如何移动两个圆盘的问题我们是已经解决了的。

也就是：红色移动到A，绿色移动到C，再把A上的红色圆盘移回到C。

就这样，三个圆盘的问题我们已经解决。

圆盘的个数不妨再加一个，这一次我们来研究四个圆盘的情况。



直接再像原来一样一步步思考如何移动？

其实这里已经没这个必要了：

在之前我们已经解决了三个圆盘的移动问题。

想一想刚刚我们在两个地方“停一下”，忘记了可以再回去看看。

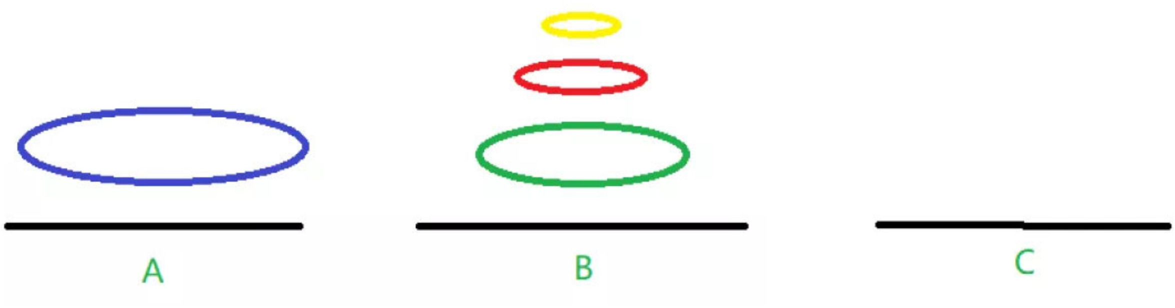
类似于刚刚的情况，我们可以这样处理：

先把A最上面的三个圆盘移动到B，再把A最下面的圆盘移动到C，最后再把已经在B上的三个圆盘移动到C。

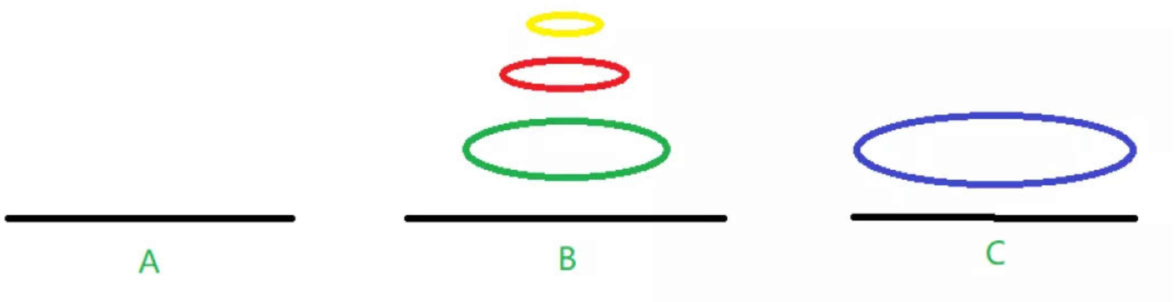
就这样，整个问题就解决了。

也就是，我们把整个过程分为三步：

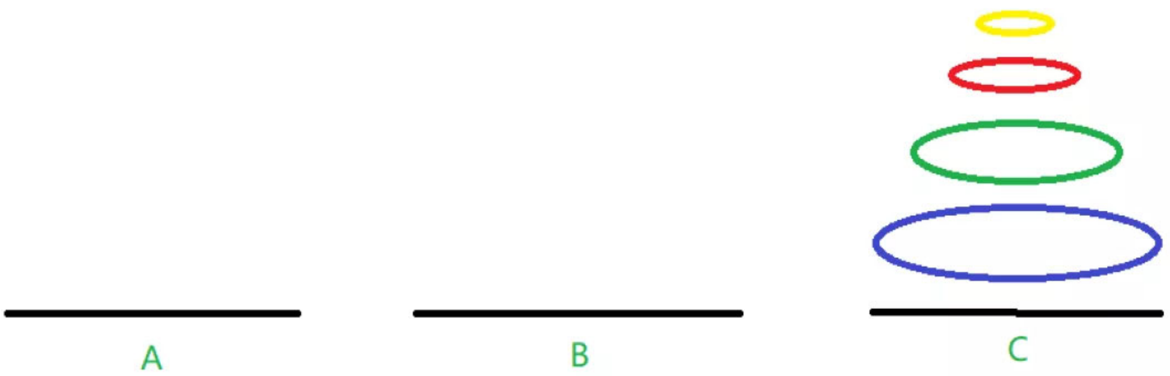
第一步：



第二步：



第三步：



至于具体每一步内的操作如何实现，我们在之前的分析都已经解决了。

也就是，

移动4个圆盘时需要用到移动3个圆盘的操作，

移动3个圆盘时需要用到移动2个圆盘的操作，

移动2个圆盘时需要用到移动1个圆盘的操作，

移动1个圆盘只需要直接一步。

以此类推，对于N个圆盘从A移动到C的问题，

我们只需要先把最上面的$N-1$个圆盘移动到B，

再把最后一个圆盘移动到C，

最后再把在B上暂存的$N-1$个圆盘移到C上就可以了。

至于$N-1$个圆盘怎样移动（从A移动到B），

只需要向前类推，

研究$N-2$个圆盘的情况，

再向前依次类推，最终推至1个圆盘的情况，

从而问题得到最终解决。

代码如下：

```
#include <iostream>
using namespace std;
void Hanoi(int n, char a, char b, char c) {
    if(n==1)
        cout<<a<<"----->"<<c<<endl;
    else {
        Hanoi(n-1,a,c,b);
        Hanoi(1,a,b,c);
        Hanoi(n-1,b,a,c);
    }
}
int main() {
    int left;
    cout<<"请输入A座上盘子的数目：";
    cin>>left;
    cout<<"将A座上的"<<left<<"个盘子全部移动到C的步骤为："<<endl;
    Hanoi(left,'A','B','C');
    return 0;
}
```

需要注意的是，

```
Hanoi(int n, char a, char b, char c)
```

这一函数的参数中，

n指需要移动的圆盘数量，

a表示起始位置对应的大写字母，

c表示目标位置对应的大写字母。

举个例子：

```
Hanoi(n-1,'A','C','B')
```

这样一行就代表把$n-1$个圆盘从A移动到B。

```
Hanoi(n-1,'B','A','C');
```

这样一行就代表把$n-1$个圆盘从B移动到C。

递归函数的核心代码只有这三行：

```
Hanoi(n-1,a,c,b);
Hanoi(1,a,b,c);
Hanoi(n-1,b,a,c);
```

也就是刚刚我们所划分的三大步。

具体是哪三步，在这里就不再重复了。

具体到函数的内部实现，也就是：

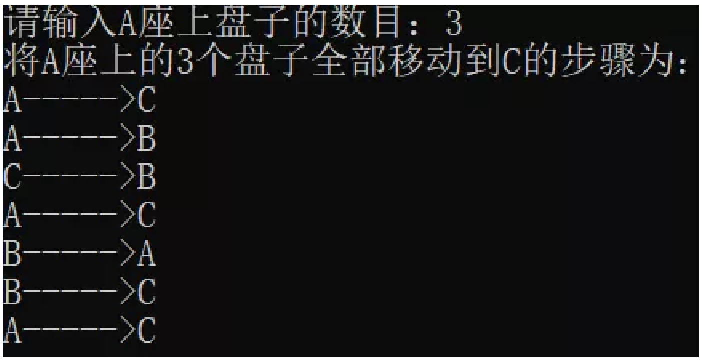
函数再次调用函数自身，

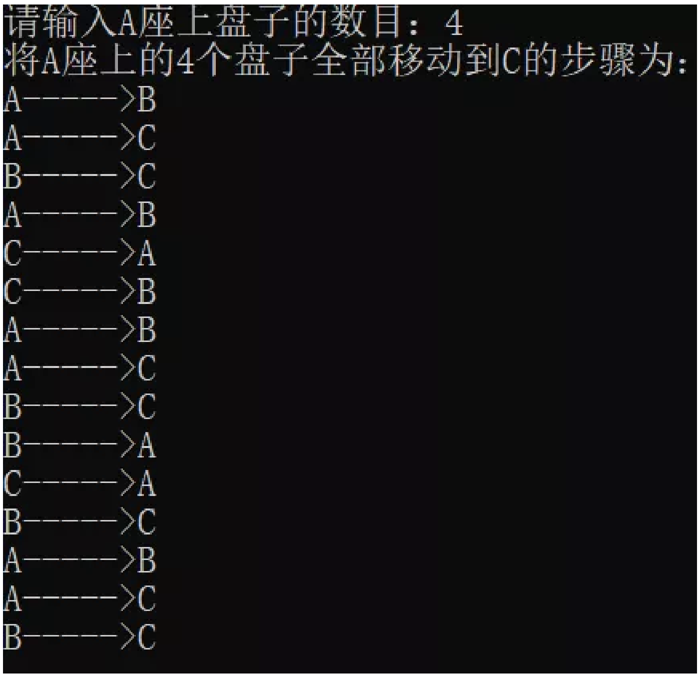
继而依次调用一直到$n = 1$，

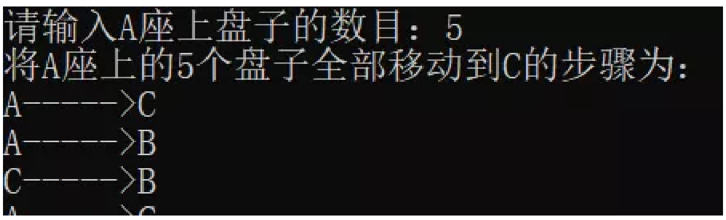
把我们的三大步依次“分解”为三个步骤，

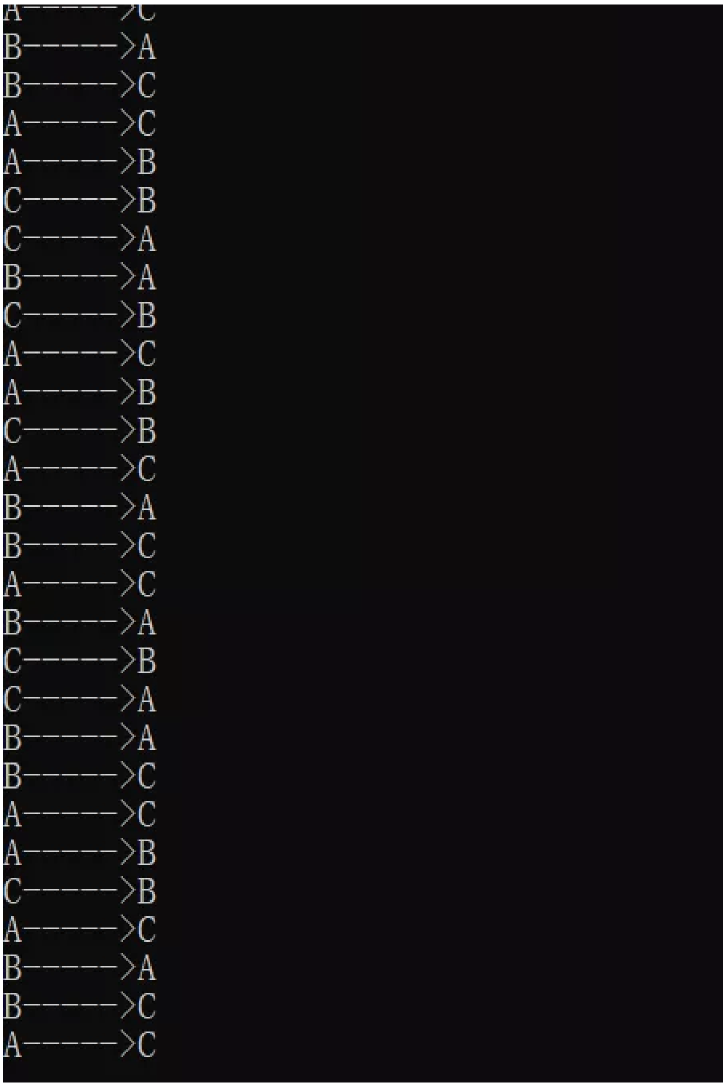
最终归于$n = 1$的情况，直接解决。

代码敲好了，我们来运行试一下：









我们可以看到，仅仅是移动五个圆盘，

就已经需要这么多步了！

如果我们自己一步一步分析，

那会有多么困难！

感谢“递归”，让复杂的汉诺塔问题变得简单。

本篇博客首发于CSDN:

https://blog.csdn.net/qq_41112170/article/details/80725336

喜欢此内容的人还喜欢

LOA公众号关闭通知

LOA算法学习笔记

我是一名美团小妹，我用我所识告诉你，城市的黑暗与光明

Cheesa黄梓暄扶绥站

李铁接近下课，国足选帅之误的启示
足球资讯速递