# 算法思想篇（递归&分治&回溯）

1.微信分销系统中有一个返利，大家应该都知道，比如B是A的下线，C是B的下线，那么在分钱返利的时候A可以分B，C的钱，这时候我们是不是就要分别找B,C的最后上级。这个问题我们一般怎么来解决呢？

A->B->C

2.斐波那契数列：

1 1 2 3 5 8 13 21

有什么特点？

从第三个数开始 就等于前面两个数相加；

数论思想：利用数学公式或者定理或者规律求解问题；

算法思想中最难的点：递归+动态规划（可以不用太懂）；树论：二叉树，红黑树

1.递归的定义：

递归是一个非常重要的算法思想，应用也是相当的广泛，包括我们后面学的数据结构尤其是树形结构里面跟递归是密不可分的。所以大家是一定要学懂它的，其实递归说起来很简单，生活中也是经常可以碰到这个场景，

比如我们在某窗口排队人太多了，我不知道我排在第几个，那么我就问我前面的人排第几个，

因为知道他排第几我就知道我是第几了。但前面的人也不知道自己排第几那怎么办呢？他也可以继续往前面问，直到问到第一个人，然后从第一个人一直传到我这里 我就很清楚的知道我是第几了。以上这个场景就是一个典型的递归。我们在这个过程中大家有没有发现一个规律那么就是会有一个问的过程，问到第一个后有一个回来的过程吧。这就是递（问）加归（回）。那么这个过程我们是不是可以用一个数学公式来求解呢？那这个数学公式又是什么？

f(n)=f(n-1)+1

f(n):表示我的位置

f(n-1):表示我前面的那个人；

自己调用自己；

2.什么样的情况下可以用递归？

（1）一个问题的解可以分解为几个子问题的解：子问题，我们通过分治的思想可以把一个数据规模大的问题，分解为很多小的问题。

我们可以把刚刚那个问前面的那个人看为子问题。大化小

（2）这个问题与分解之后的子问题，求解思路完全一样：

（3）一定有一个最后确定的答案，即递归的终止条件：刚刚那个问题就是第一个人。第一个人是肯定知道自己排第几吧即n=1的时候，如果没有这个特性那么我们这个递归就会出现死循环，最后程序就是栈溢出；stack out of

3.递归如何实现？里面有哪些算法思想？

递归，回溯；

递归的关键相信大家已经知道了就是要求出这个递归公式，找到终止条件。现在我们可以回到课堂前跟大家讲的那个斐波那契数列数列：

1 1 2 3 5 8 13 这个的数列我们称之为斐波那契数列

他的求解公式：f(n)=f(n-1)+f(n-2)

终止条件：n<=2 f(n)=1

4.递归的时间和空间复杂度分析：

以斐波那契数列为例为分析递归树：

f(n)=f(n-1)+f(n-2)

分析一段代码好坏，有两个指标，时间复杂度和空间复杂度 都是：O(2^n)=>O(n)或者O(nlogn)

5.递归优化：

(1)使用非递归。所有的递归代码理论上是一定可以转换成非递归的。

(2)加入缓存：把我们中间的运算结果保存起来，这样就可以把递归降至为o(n)

(3)尾递归：什么是尾递归？尾递归就是调用函数一定出现在末尾，没有任何其他的操作了。因为我们编译器在编译代码时，如果发现函数末尾已经没有操作了，这时候就不会创建新的栈，而且覆盖到前面去。

倒着算，不需要在回溯了，因为我们每次会把中间结果带下去。

布置一个难的作业：把归并排序写出来；JDK的排序源码