分治法感想

每一道使用递归和分治策略的题目，在看到题解之前，我都觉得很难，没有任何的思路。在看懂题解之后，我又觉得很简单（求解的思路和数学表达）。但是，经过大学里一年半大量代码的训练，我已经比刚开始学C语言那会有了很大的进步，对于使用递归和分治策略的题目，也能够逐渐找到它的递归结构和出口条件。如果想要熟练地使用这种策略解决问题，还需要大量的练习。

分治法的基本思想是将一个规模为n的问题分解为k个规模较小的子问题，这些子问题相互独立且与原问题相同。递归地（多数情况）解这些子问题，然后将子问题的解合并得到原问题的解。一般的算法设计模式如下。

```
divide-and-conquer(P)
{
    if(|P|<=n0)adhoc(P);    //问题规模小于阈值，直接求解
    divide P into smaller subinstances P1, P2, P3, ...,Pk;  //将问题分成k个规模相同的子问题
    for(i=1,i<=k,i++)
        yi = divide-and-conquer(Pi);    //逐个解决k个子问题
    return merge(y1, ..., yk);  //合并k个子问题的解，得到原问题的解
}
```

从上面的伪代码中可以看出，分治法的两个最重要的步骤就是划分与合并。

分治法的实现往往需要使用到递归，在学习的过程中，我对于递归和分治的联系产生了一些观点。

- 区别：
    1. **递归的子问题不是独立的，原问题的解包含着子问题的解。**比如全排列问题，n个元素的全排列包含着n-1个元素的全排列，要想解决n个元素的全排列问题，必须先解决规模为n-1的全排列问题。**然而，分治法的子问题是规模相同，相互独立，相互正交的。**比如归并排序，对前半个数组排序和对后半个数组排序这两个子问题互不干扰，相互独立。
    2. **递归求解的顺序是唯一的。**比如Hanoi塔问题，必须先解决了n-1层的Hanoi塔，才能解决n层的Hanoi塔。**然而，分治法的求解顺序可以是任意的。**比如棋盘覆盖问题，左上、右上、左下、右下四个子问题可以任意指定它们的求解顺序。
    3. **传统的（典型意义上的）递归，每次递归的子问题的规模只是原问题规模减小一个常数的级别。**如Fibonacci数列，它的递推公式

       $$F(n) = F(n-1) + F(n-2), n > 1$$

       只涉及到n-1和n-2。**但是，分治法的子问题规模是原问题规模除以一个常数。**棋盘覆盖子问题的规模是原问题除以4，归并排序就是原问题除以2。当然也有特例，比如快速排序，如果基准元素选取的好，那么子问题的规模就是原问题除以2，如果基准元素选的不好，那么子问题还是n-1的规模。
- 联系：
    1. 分治法的求解需要使用到递归，也就是说分治法求解的函数也需要自己调用自己。
    2. 分治法和递归在问题是小规模的时候都很好解。

分治法的一些特点如下：

1. 小规模的问题好解决
2. 满足最优子结构性质（原问题的解包含子问题的解）
3. 子问题的规模差不多大（问题最好是2^n的规模）

4. 子问题相互独立
5. 合并解简单

分治法对于求解复杂问题是很有优势的。分治法孕育了计算机科学中许多最重要和最有效的算法。对于并行计算是非常理想的，因为各个子问题都可以由不同的处理机同时计算。