递归与分治策略

分治法的适用条件

分治法所能解决的问题一般具有以下几个特征：

- 该问题的规模缩小到一定的程度就可以容易地解决；

因为问题的计算复杂性一般是随着问题规模的增加而增加，因此大部分问题满足这个特征。

分治法的适用条件

分治法所能解决的问题一般具有以下几个特征:

- 该问题的规模缩小到一定的程度就可以容易地解决;
- 该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质

这条特征是应用分治法的前提，它也是大多数问题可以满足的，此特征反映了递归思想的应用

分治法的适用条件

分治法所能解决的问题一般具有以下几个特征：

- 该问题的规模缩小到一定的程度就可以容易地解决；
- 该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质
- 利用该问题分解出的子问题的解可以合并为该问题的解；

能否利用分治法完全取决于问题是否具有这条特征，如果具备了前两条特征，而不具备第三条特征，则可以考虑**贪心算法**或**动态规划**。

分治法的适用条件

分治法所能解决的问题一般具有以下几个特征：

- 该问题的规模缩小到一定的程度就可以容易地解决；
- 该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质
- 利用该问题分解出的子问题的解可以合并为该问题的解；
- 该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子问题。

这条特征涉及到分治法的效率，如果各子问题是不独立的，则分治法要做许多不必要的工作，重复地解公共的子问题，此时虽然也可用分治法，但一般用**动态规划**较好。

分治法的基本步骤

```
divide-and-conquer(P)
 {
   if ( | P | <= n0) adhoc(P);   //解决小规模的问题
   divide P into smaller subinstances P1,P2,...,Pk;  //分解问题
   for (i=1,i<=k,i++)
     yi=divide-and-conquer(Pi);  //递归的解各子问题
   return merge(y1,...,yk);  //将各子问题的解合并为原问题的解
 }
```

平衡子问题

分治法的复杂性分析

设问题规模足够小的adhoc耗费1个单位时间。

设分解子问题及子问题的解合并需用f(n)个单位时间。

用T(n)表示该分治法解规模为|P|=n的问题所需的计算时间，则有：

$$T(n)=\begin{cases} O(1) & n=1 \\ kT(n/m)+f(n) & n>1 \end{cases}$$

求得方程的解：

$$T(n)=n^{\log_m k}+\sum_{j=0}^{\log_m n-1}k^j f(n/m^j)$$

分治法的复杂性分析

设分解阀值n_0=1，且adhoc解规模为1的问题耗费1个单位时间。设将原问题分解为k个子问题以及用merge将k个子问题的解合并为原问题的解需用f(n)个单位时间。
用T(n)表示该分治法解规模为|P|=n的问题所需的计算时间，则有：

$$T(n)=\begin{cases} O(1) & n=1 \\ kT(n/m)+f(n) & n>1 \end{cases}$$

通过迭代法求得方程的解：$T(n)=n^{\log_m k}+\sum_{j=0}^{\log_m n-1} k^j f(n/m^j)$

注意：递归方程及其解只给出n等于m的方幂时T(n)的值，但是如果认为T(n)足够平滑，那么由n等于m的方幂时T(n)的值可以估计T(n)的增长速度。通常假定T(n)是单调上升的，从而当$m_i \le n < m_{i+1}$时，$T(m_i) \le T(n) < T(m_{i+1})$。

如何求解具有递归关系的复杂度?