分治

如何降低递归算法复杂度

- 代数变换减少子问题个数
  - 例子:
    位乘问题：设 X，Y 是两个 n 位二进制数，$n=2^k$，求 XY
    - 若采用一般分治算法:

$$
\begin{aligned}
&X=A2^{n/2}+B,\ \ Y=C2^{n/2}+D\\
&XY=AC2^n+(AD+BC)2^{n/2}+BD\\
&W(n)=4W(n/2)+cn,\ \ W(1)=1\\
\Rightarrow &W(n)=O(n^{log4})=O(n^2)
\end{aligned}
$$

    - 采用代数变换:

$$
\begin{aligned}
&AD+BC=(A-B)(D-C)+AC+BD\\
now\ \ &W(n)=3W(n/2)+cn\\
z&W(1)=1\\
\Rightarrow &W(n)=O(n^{log3})=O(N^{1.59})
\end{aligned}
$$

- 预处理减少递归的操作
  - 例子
    平面最近点对算法：对于平面上 n 个点的集合 P，求最近的两个点及其距离
    - 采用一般递归算法:
      - 先把所有点按照 X，Y 排序
        $O(nlogn)$
      - 然后作垂线将点集分为左右两个规模相等的部分 P_L，P_R
        $O(1)$
      - 递归求出左右两边的最近点对，将两边的最近距离对比求出所有点的最近距离 δ。
        $2T(n/2)$
      - 对于在垂直线两边距离 δ 范围内的每个点，检查是否有点与它的距离现于 δ 如果存在则将 δ 修改为新值。
        $O(n)$
      - 递推方程:

$$
\begin{aligned}
&T(n)=2T(/2)+O(nlogn)\\
\Rightarrow &T(n)=O(nlog^2n)
\end{aligned}
$$

    - 预处理提前排好序:

$$
\begin{aligned}
&T(n)=2T(n/2)+O(n)\\
\Rightarrow &T(n)=O(nlogn)
\end{aligned}
$$

典型案例

- 找第 K 大的数:
  - 找最大和最小值:
    分治算法：
    - 将 n 个元素两两一组分成 $\lfloor n/2\rfloor$ 组，
    - 分别求各组内的最大值和最小值
      $O(\rfloor n/2\lfloor)$
    - 最后在 $\lceil n/2\rceil$ 个最大值中直接比较选出最大值，在 $\lceil n/2\rceil$ 个最小值中直接比较选出最小值 。
      $O(2\lceil n/2\rceil-2)$
    - 复杂度:

$$W(n) = \lceil 3n/2 \rceil - 2$$

- 找第二大：
  分治算法：
  - 将所有元素两两一组
  - 每组两个元素比较大小，将较小的元素记录在较大元素指向的链表中
  - 最后在最大值的链表中直接比较找最大值
  - 复杂度：

$$W(n) = n - 1 + \lceil logn \rceil - 1 = n + \lceil logn \rceil - 2$$

- 找第 k 小的数
  分治算法 $Select(S, k)$：首先类似于快速排序的做法选出一个轴值，并将这个轴值放在正确的位置，设为第 q 个数。如果 q = k 则这个数就是所求值，若 q > k 则在轴值左侧较小的元素中递归求第 k 小的值，若 q < k 则在轴值右侧较大的元素中递归求第 k-q 小的元素。下面的算法唯一的改进在于轴值的选取让划分更均匀。
  - 将所有数划分为 5 个一组，共 $n_M = \lceil n/5 \rceil$ 个组
  - 每组找中位数，n_M 个中位数构成集合 M
  - 递归调用 $Select(M, \lceil |M|/2 \rceil)$ 求出 M 的中位数 m^*。
  - 中位数小于 m^* 的五元组中 **小于** 其中位数的数必定小于 m^*, 中位数大于 m^* 的五元组中 **大于** 其中位数的数必定大于 m^*。将中位数小于 m^* 的五元组中 **大于** 其中位数的数和中位数大于 m^* 的五元组中 **小于** 其中位数的数逐个与 m^* 比较，最终找出所有大于 m^* 的数和所有小于 m^* 的数。
  - 这时可以将所有元素分成值较大和较小的两组，并按照前述方法完成选择。