作业一讲评　　2013.10.29

1.14 函数的阶

$$5\log(n+100)^{10} = \Theta(\log n),$$

$$1+\frac{1}{2}+\ldots+\frac{1}{n} = \Theta(\log n),$$

$$(\ln n)^2 = \Theta(\log^2 n), \quad \sqrt[3]{n}+\log n = \Theta(\sqrt[3]{n}),$$

$$\log n^{n+1} = \Theta(n\log n), \quad \log(n!) = \Theta(n\log n),$$

$$0.001n^4+3n^3+1 = \Theta(n^4), \quad 3^n = \Theta(3^n),$$

$$2^{2n} = \Theta(4^n), \quad (n-2)! = \Theta((n-2)!)$$

1.16 (2) $T(n)=9T(n/3)+n$, $T(1)=1$

(6) $T(n)=2T(n/2)+n^2\log n$, $T(1)=1$

(2) 使用主定理，$a=9, b=3, f(n)= n$, 因此

$$T(n)=O(n^{\log_3 9})=O(n^2)$$

(6) 使用主定理. $a=2, b=2$, $f(n)=n^2\log n$, $f(n)=\Omega(n)$. 取$c=3/4$，则

$$af(\frac{n}{b})=2(\frac{n}{2})^2\log(\frac{n}{2})=\frac{n^2}{2}(\log n-1)$$

$$\le\frac{n^2}{2}\log n\le cn^2\log n=cf(n)$$

于是$T(n)=\Theta(n^2\log n)$

2.5 找 x 使得 $L<x<U$

在 A 中使用二分查找算法找 L.

如果 $L=A[i]$，找到 L 位置 i，然后把 i 加1；

如果 L 不在 A 中，找到大于 L 的最小数的位置 i.

类似地，在 A 中二分查找 U.

找到 U 所在的位置 j，然后把 j 减1；

如果 U 不在 A 中，找到小于 U 的最大数的位置 j.

输出 A 中从 i 到 j 的全体数.

算法的时间是 $O(\log n)$.

2.7 主元素测试（可排序）

算法1:

对A排序；顺序检索A，计数每个元素 x 的个数. 保留数目最多的元素 x 及其个数 j. 检索完成后如果 $j>n/2$，x就是主元素，否则A中没有主元素.

$$T(n)=O(n\log n)+O(n)=O(n\log n)$$

算法2：找出A的中位数x, 顺序比较计数 x 在A中出现的次数 j. 如果 $j>n/2$，则 x 就是主元素，否则不存在主元素.

$$T(n)=O(n)+O(n)=O(n)$$

命题：如果 x 是主元素，则 x 是中位数.

主元素测试（不可排序）

算法3：芯片测试算法

淘汰过程：将元素两两分组进行比较，如果相等则选一个进入下一轮，否则两个都淘汰. 轮空元素需特殊处理.

命题：如果存在主元素，测试后一定会留下来

性质

1. 每轮测试后输入至少减半，测试至多**log*n***次
2. 中间被淘汰不是主元素. 如果最后有元素留下来，测试该元素在数组中的个数，即可做出判断

时间： $O(n)$

主元素测试：栈（不可排序）

设计思想：

1. 设立栈 S，大小为$n/2$. 依次检查 A 中元素 x
2. 如果栈为空或者 x=栈顶元素，x进栈；否则弹出栈顶元素.
3. 全部检查完成后，若栈为空，回答“No”；如果有元素，则计数该素在A中出现的次数，当次数>$n/2$时回答“yes”否则“No”

命题：

1．如果 x 为主元素，则 x 一定保留在栈里

2．栈中只有1 种元素

时间：$O(n)$

2.18 与原点最近的某些点

算法 **FindPoint (A,B)**

输入：$A=\{x_1, x_2, \ldots, x_n\}$，$B=\{y_1, y_2, \ldots, y_n\}$

输出：距离原点最近的 $\lfloor\sqrt{n}\rfloor$ 个点的标号

1. **for** $i \leftarrow 1$ **to** n **do** $d_i \leftarrow \sqrt{x_i^{\,2} + y_i^{\,2}}$
2. $k \leftarrow \lfloor\sqrt{n}\rfloor$
4. 在$\{d_1,d_2,\ldots,d_n\}$中找第k小的元素b
5. 比较d_i与b，把 k个小于等于b 的 d_i 及其下标 i 存放到C中
6. C中d_i 按照从大到小顺序排序(移动d_i时下标i同时移动).
7. **for** $j \leftarrow 1$ **to** k **return** $C[j]=d_i$ 的下标 i.

$$W(n) = O(n) + O(k \log k)$$

$$= O(n) + O(\sqrt{n} \log \sqrt{n}) = O(n)$$

2.21 选第$k_1, k_2, \ldots, k_r$小

解 (1) 依次调用选择算法 Select(S,k_i), i=1,2,…,r. 每次调用最坏情况下的时间复杂度为$O(n)$，总共调用r次，因此最坏情况下的时间复杂度为$O(nr)$.

(2) 当r>1时采用分治策略，假设原始输入是(S,K)，其中$K=\{k_1, k_2, \ldots, k_r\}$. 取$t=\lceil r/2 \rceil$，调用Select算法计算 S 的第 k_t小x. 用x将S划分成不超过 x 的数的集合 S_L和大于x 的数的集合S_R. 从而将原问题归约为(S_L, K_L)和(S_R,K_R)两个r减半的子问题，其中$K_L=\{k_1,k_2,\ldots,k_{t-1}\}$, $K_R=\{k_{t+1},\ldots,k_r\}$. 递归求解$(S_L, K_L)$和$(S_R,K_R)$即可.

2.21（续）

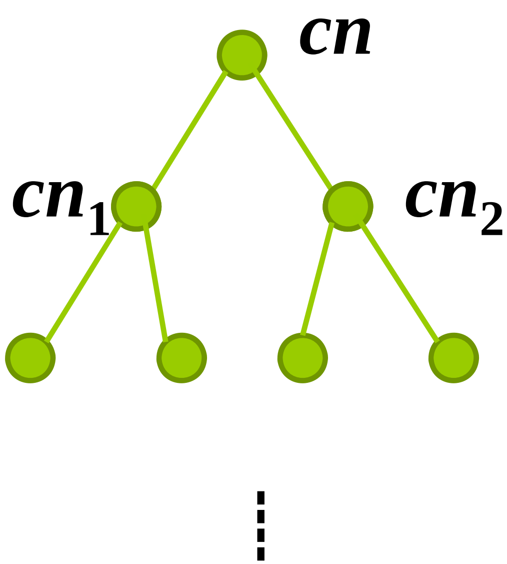
设原始问题(S,K)的规模为n和r，其中$n=|S|$，$r=|K|$. 经过归约得到两个子问题，其中(S_L,K_L)的规模为n_1, $r/2$, (S_R,K_R)的规模为n_2,$r/2$. 令$T(n,r)$是对(S,K)的工作量，那么有

$$T(n,r)= T(n_1,r/2)+T(n_2,r/2)+O(n) \qquad r>1$$

$$T(n,1)=O(n)$$

上述递推式中的$O(n)$包含求S第k_t小的数x的工作量以及用x划分S的工作量. 用递归树求解. 不难看出，在递归树的每一层结点上的工作量之和都等于$O(n)$，总计$\log r$层，因此总工作量$T(n,r)=O(n\log r)$.

2.21 (续)



cn $\quad$ $r_0=r$

$cn_1+cn_2=cn$ $\quad$ $r_1=r/2$

cn $\quad$ $r_2=r/4$

⋮

cn $\quad$ $r_k=1$

$k=O(\log r)$