第八次书面作业评分标准

8.2 对于给定的 $x\neq 0$，求 n 次多项式 $P(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n$ 的值.

(1) 设计一个在最坏情况下时间复杂度为 $\Theta(n)$ 的求值算法.

(2) 证明任何求值算法的时间复杂度都是 $\Omega(n)$.

第一问10分，第二问10分。 大致思路对即可，其余情况酌情扣分

8.2 (1) 迭代计算：

$$\begin{aligned}P_1(x)&=a_n\\P_2(x)&=a_{n-1}+P_1(x)\cdot x\\P_3(x)&=a_{n-2}+P_2(x)\cdot x\\&\vdots\\P_{n+1}(x)&=a_0+P_n(x)\cdot x\end{aligned}$$

不难看出

$$P_{n+1}(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n=P(x)$$

如果顺序计算多项式 $P_1(x),P_2(x),\cdots,P_{n+1}(x)$，最后得到的就是 $P(x)$. 在这个过程中，计算 $P_i(x)$ 需要用到 $P_{i-1}(x)$ 的值，而且只需要 1 次乘法和 1 次加法，是常数时间. 于是计算整个多项式序列仅需要 $\Theta(n)$ 时间.

(2) **证** 多项式 $P(x)$ 有 $n+1$ 个系数，每个系数至少需要被处理 1 次. 如若不然，假设某个求值算法 A 不对系数 a_i 进行处理. 考虑输入

$$<a_0,a_1,\cdots,a_{i-1},a_i,a_{i+1},\cdots,a_n> \text{ 和 } <a_0,a_1,\cdots,a_{i-1},b_i,a_{i+1},\cdots,a_n>$$

其中 $a_i\neq b_i$ 是不为 0 的数，其他系数都相等. 那么这两个多项式对同一个 x 的求值一定不等，但是算法 A 的输出是不加区别的. 因此算法 A 出错. 于是任何算法至少需要 $n+1$ 次运算，即在最坏情况下时间复杂度为 $\Omega(n)$.

8.4 设 n 是 k 的倍数，有 k 个排好序的数表 $L_1, L_2, \cdots, L_k$，每个数表都有 n/k 个数. 假设 n 个数彼此不等，并且归并长为 m, n 的两个数表的时间代价是 $O(m+n)$.

(1) 使用顺序归并算法归并这 k 个数表，在最坏情况下的时间复杂度是什么？

(2) 设计一个时间复杂度更低的归并算法，说明算法的主要设计思想，并分析你的算法在最坏情况下的时间复杂度.

(3) 对于以比较作为基本运算求解上述问题的算法类，最坏情况下的时间复杂度的下界是什么？证明你的结果.

(1)说明最坏情况给5分，分析正确复杂度5分

(2)设计算法5分，然后分析复杂度给5分

(3)构造决策树给8分，分析正确结果给2分

其余情况酌情扣分

8.4 (1) 顺序归并算法. 伪码如下：

1. $L \leftarrow L_1$
2. for $j \leftarrow 2$ to k
3. $\quad L \leftarrow L \cup L_j$ $\quad$ //L 与 L_j 归并得到新的 L
4. return L

在归并过程中 L_1 中的元素被比较 $k-1$ 次，L_2 的元素被比较 $k-1$ 次，$\cdots$，L_{k-1} 的元素被比较 2 次，L_k 的元素被比较 1 次. 总共比较

$$n/k[(k-1)+(k-1)+(k-2)+\cdots+2+1] = n(k^2+k-2)/2k = O(kn)$$

(2) 使用二分归并算法. 伪码如下：

1. $l \leftarrow k$
2. 将 l 个表两两一组，分成 $l/2$ 组
3. 每组的 2 个表进行归并
4. if $\quad l$ 为奇数
5. then $l \leftarrow l/2 + 1$
6. else $l \leftarrow l/2$
7. if $\quad l > 1$ then goto 2
8. return L

最坏情况的时间复杂度：每个表的元素被归并 $\log k$ 次，每次归并比较 1 次. 因此总的时间 $W(n) = O(n\log k)$.

(3) 构造决策树如下：

1. 如果算法比较元素 a_i 和 a_j，那么将结点标记为(i,j)；

2. 结点被标记(i,j)之后，

若 $a_i<a_j$，下一步算法比较 a_k 与 a_l，那么(i,j)的左儿子标记为(k,l)；若比较后算法结束，用输入把左儿子标记为树叶.

若 $a_i>a_j$，下一步算法比较 a_k 与 a_l，那么(i,j)的右儿子标记为(k,l)；若比较后算法结束，用输入把右儿子标记为树叶.

将 n 个数分成 k 组，每组 n/k 个数，先选 L_1 中的 n/k 个数，然后从剩下的 $n-n/k$ 个数中选 L_2 中的数，…，从 $n-(k-1)n/k$ 个数中选 n/k 个数. 因此不同的输入个数为 $n!/[(n/k)!]^k$，在决策树中对应了所有的树叶. 给定一个输入，从树根开始，通过比较运算沿着一条路径走到树叶，那么决策树的深度代表了最坏情况下的时间复杂度，树深

$$d=\left\lceil \log\frac{n!}{[(n/k)!]^k}\right\rceil \geqslant \log(n!)-k\log((n/k)!)$$

$$=\Theta(n\log n-k(n/k)(\log n-\log k))=\Theta(n\log k)$$

二分归并算法是最优的算法.

8.8 设 S 是 n 个数构成的数组，判断 S 中的元素是否都是唯一的. 如果唯一，则输出“Yes”；否则输出“No”. 证明唯一性判定问题的复杂度是 $\Theta(n\log n)$.

15分，给出决策树5分，然后证明下界5分，证明紧的界给5分，其余情况酌情扣分。

8.8 证 设输入是多重集 $S=\{n_1\cdot a_1,n_2\cdot a_2,\cdots,n_k\cdot a_k\}$，$S$ 含有 k 种元素，元素 a_i 的重复度是 n_i，$i=1,2,\cdots,k$，且 $\sum_{i=1}^{k}n_i=n$. 如下构造决策树：

1. 算法第一次比较的元素是 a_i 和 a_j，那么树根标记为(i,j).

2. 如果 $a_i=a_j$，那么算法结束，并将输入标记树叶，作为(i,j)的左儿子.

3. 如果 $a_i\neq a_j$，算法下一步比较的元素是 a_k 和 a_l，那么将(k,l)标记为(i,j)的右儿子；如果比较后算法结束，将(i,j)的右儿子作为树叶，并标记为输入.

多重集 S 的排列数为

$$\binom{n}{n_1n_2\cdots n_k}=\frac{n!}{n_1!n_2!\cdots n_k!}$$

由于存在着 n 个元素都不相等的输入，即 $n_1=n_2=\cdots=n_k=1$. 对这个输入，决策树的叶片数达到 $n!$，树的深度为 $\Theta(n\log n)$，从而证明了在最坏情况下算法时间复杂度的下界是 $\Theta(n\log n)$.

显然这是一个紧的界. 先用 $O(n\log n)$时间将 S 中的数按照从小到大的顺序排列为 a'_1，a'_2，…，a'_n，相同的数一定是连续分布的. 然后用 $O(n)$时间顺序检查每对相邻的元素 a'_i和 a'_{i+1}是否相等，总计用 $O(n\log n)$时间就可以判定 S 中的元素是否唯一.

8.9 设 $L=\{a_1,a_2,\cdots,a_n\}$ 是 n 个不相等的实数的数表，m 是小于 n 的正整数. 现在需要按照从小到大的次序输出 L 中最小的 m 个数.

(1) 如果 $m=\Theta(n/\log n)$，以 L 中元素的比较作为基本运算，设计一个 $O(n)$ 时间的算法.

(2) 如果 $m=\omega(n/\log n)$，证明不存在 $O(n)$ 时间的算法.

20分，第一问10分，给出大概的算法5分，然后证明复杂度5分。第二问10分，指出m个数排序5分，复杂度5分。其余情况酌情扣分。

8.9 (1) 在 L 中找第 m 小的数 a_i；然后把 a_i 和其他数比较，找到所有小于 a_i 的数；对所有小于 a_i 的 $m-1$ 个数进行排序；然后从小到大输出. 找 a_i 的时间是 $O(n)$，找所有比 a_i 小的数的时间是 $O(n)$，对 $m-1$ 个数排序时间是 $m\log m$，而 $m=\Theta(n/\log n)$，于是

$$\begin{aligned}T(n)&=O(n)+O(n)+O\left(\frac{n}{\log n}\log\frac{n}{\log n}\right)\\&=O(n)+O\left(\frac{n}{\log n}(\log n-\log\log n)\right)\\&=O(n)\end{aligned}$$

(2) **证** 任何算法都对 m 个数排序，时间复杂度 $T(n)=\Omega(m\log m)$，如果 $m=\omega(n/\log n)$，则

$$\begin{aligned}T(n)&=\Omega(m\log m)=\Omega\left(\omega\left(\frac{n}{\log n}\log\frac{n}{\log n}\right)\right)\\&=\omega\left(\frac{n}{\log n}(\log n-\log\log n)\right)\\&=\omega\left(n-\frac{n\log\log n}{\log n}\right)=\omega(n)\end{aligned}$$

8.10 证明任何从 n 个数中选第 k 小的数的算法，如果以比较作为基本运算，那么它至少要做 $n+\min\{k,n-k+1\}-2$ 次比较.

共15分， 给出类似于决定性比较的描述给5分，构造的输入方式正确给5分，结果的证明给5分，其余情况酌情扣分

8.10 **证** 设第 k 小的数为 t，首先定义决定性的比较与非决定性的比较. 决定性的比

较就是建立了 x 与 t 关系的比较. 例如：

$\exists y(x > y$ 且 $y \geqslant t)$，x 满足上述条件的第一次比较

$\exists y(x < y$ 且 $y \leqslant t)$，x 满足上述条件的第一次比较

当 $x>t, y<t$ 时，$x>y$ 的比较不是决定性的. 为找到第 k 小的数 t，必须要做 $n-1$ 次决定性的比较.

设 A 是找第 k 小的任意算法，针对算法 A 如下构造输入.

1. 分配一个值给第 k 小的数 t.

2. 如果 A 比较 x 与 y，且 x 与 y 没有被赋值，那么赋值 x, y 使得 $x>t, y<t$.

3. 如果 A 比较 x 与 y，且 $x>t, y$ 没被赋值，则赋值 y 使得 $y<t$.

4. 如果 A 比较 x 与 y，且 $x<t, y$ 没被赋值，则赋值 y 使得 $y>t$.

5. 令 $c=\min\{k-1, n-k\}$，如果存在 c 个元素已得到小于 t 的值，则对未赋值的全部分配大于 t 的值.

6. 如果存在 c 个元素已得到大于 t 的值，则对未赋值的全部分配小于 t 的值.

7. 如果剩下 1 个元素则分配 t 给它.

这样赋值的输入使得 A 在这个输入下所进行的上述比较都是非决定性的. 这样的比较至少有 $c=\min\{k-1, n-k\}$ 个. 因此总比较次数至少为

$$n-1+c = n-1+\min\{k-1, n-k\} = n+\min\{k, n-k+1\}-2$$