算法设计与分析第四次作业参考答案

答案仅供参考，合理即酌情给分

整体评分思路：重点是贪心策略是否写错，以及是否使用交换论证或归纳法给出证明，证明过程言之有理即可。最后检查伪代码是否正确。

4.1：15%（贪心策略 3%，交换论证 5%，伪代码算法描述 5%，计算解的结果 2%）

4.1 （1）假设调度 f 的顺序是 $i_1,i_2,\cdots,i_n$，那么 i_k 的等待时间是 $\sum_{j=1}^{k-1} t_{i_j}$. 总的等待时间

$$time(f)=\sum_{i=1}^{n}(n-i)t_{i_j}$$

贪心策略是：服务时间较短的优先安排. 排序使得 $t_1 \leqslant t_2 \leqslant \cdots \leqslant t_n$，按照 $1,2,\cdots,n$ 的顺序安排服务. f^* 的总等待时间为

$$time(f^*)=\sum_{i=1}^{n}(n-i)t_i$$

命题 4.1 对任何输入，对服务时间短的顾客优先安排将得到最优解.

证 使用交换论证的方法. 假设存在某个最优解 g 存在 $i,j\in A$，$i<j$，但是 i 在 j 之后得到服务. 那么在 g 的安排中一定存在相邻安排的顾客 i 和 j，使得 $i<j$ 但 i 在 j 之后得到服务. 交换 i 和 j 得到新的服务顺序 g'. 那么

$$time(g)-time(g')=t_j-t_i \geqslant 0$$

总等待时间将不会增加，因此 g' 也是最优解. 至多经过 $n(n-1)/2$ 次这样的交换，就可以将 g 转化成算法的解 f^*. 从而证明了 f^* 是最优解.

（2）算法 Service(T).

输入：$T[1..n]$ 是服务时间 $t_1,t_2,\cdots,t_n$ 的数组

输出：函数 f，$f(i)$ 为第 i 个顾客的开始服务时刻，$i=1,2,\cdots,n$

```
1.  sort(T)                    //按照服务时间从小到大的顺序排列
2.  f(1)←0
3.  for i←2 to n
4.      f(i)←f(i-1)+t_{i-1}
5.  return f
```

算法的最坏情况下的时间复杂度为 $O(n\log n)$.

（3）服务顺序为：1,3,2,6,5,7,4. 其开始服务时刻为：$f(1)=0$，$f(3)=1$，$f(2)=3$，$f(6)=6$，$f(5)=12$，$f(7)=22$，$f(4)=34$. 总等待时间

$$Time(f)=1\times 6+2\times 5+3\times 4+6\times 3+10\times 2+12\times 1=78$$

4.3：15%（贪心策略：3%，伪代码算法描述 5%，算法证明 7%）

4.3 使用贪心法．令 $a_1, a_2, \cdots$ 表示基站的位置．

贪心策略：首先令 $a_1 = d_1 + 4$．对 $d_2, d_3, \cdots, d_n$ 依次检查，找到下一个不能被该基站覆盖的房子．如果 $d_k \leqslant a_1 + 4$ 但 $d_{k+1} > a_1 + 4$，那么第 $k+1$ 个房子不能被基站覆盖，于是取 $a_2 = d_{k+1} + 4$ 作为下一个基站的位置．照此下去，直到检查完 d_n 为止．

算法的伪码如下：

Location

输入：距离 $d_1, d_2, \cdots, d_n$ 的数组 $d[1..n]$，满足 $d[1] < d[2] < \cdots < d[n]$

输出：基站位置的数组 a

```
1. a[1]←d[1]+4
2. k←1
3. for j←2 to n
4.     if d[j]>a[k]+4
5.     then k←k+1
6.          a[k]←d[j]+4
7.   return a
```

算法正确性证明使用归纳法．

命题 4.3 对任何正整数 k，存在最优解包含算法前 k 步选择的基站位置．

证 $k=1$，存在最优解包含 $a[1]$．若不然，有最优解 OPT，其第一个位置是 $b[1]$，$b[1] \neq a[1]$，那么 $d_1 - 4 \leqslant b[1] < d_1 + 4 = a[1]$．$b[1]$覆盖的是距离在$[d_1, b[1]+4]$之间的房子．$a[1]$覆盖的是距离在$[d_1, a[1]+4]$的房子．因为 $b[1] < a[1]$，$b[1]$覆盖的房子都在 $a[1]$覆盖的区域内，用 $a[1]$替换 $b[1]$，得到的仍旧是最优解．

假设对于 k，存在最优解 A 包含算法前 k 步选择的基站位置，即

$$A = \{a[1], a[2], \cdots, a[k]\} \cup B$$

其中 $a[1], a[2], \cdots, a[k]$覆盖了距离 $d_1, d_2, \cdots, d_j$ 的房子．那么，B 是关于 $L = \{d_{j+1}, d_{j+2}, \cdots, d_n\}$的最优解．否则，存在关于 L 的更优的解 B^*，那么用 B^* 替换 B 就得到 A^*，且$|A^*| < |A|$，与 A 的最优性矛盾．根据归纳基础，L 有一个最优解 $B' = \{a[k+1], \cdots\}$，$|B'| = |B|$．于是

$$\begin{aligned} A' &= \{a[1], a[2], \cdots, a[k]\} \cup B' \\ &= \{a[1], a[2], \cdots, a[k], a[k+1], \cdots\} \end{aligned}$$

且$|A'| = |A|$，A'也是最优解．从而证明了命题对 $k+1$ 也为真．根据归纳法，对任何正整数 k 命题都成立．

第 3 行的 for 循环运行 $O(n)$次，循环体内操作为常数时间，因此算法最坏情况下的时间复杂度是 $O(n)$．

4.6 15%（贪心策略 5%，算法证明 10%）

4.6 使用贪心法.

贪心策略：按照加工时间从小到大对作业排序，使得 $t(i) \leqslant t(i+1), i=1,2,\cdots,n-1$. 按照标号从小到大的次序安排所有的作业.

命题 4.6 对任何输入，按照加工时间从小到大安排任务将得到最优解.

证 使用交换论证的方法. 任意给定最优解 $f=<i_1,i_2,\cdots,i_n>$，不妨设从 0 时刻计时，且 f 的安排中没有空闲时间. 任务 i_j 的完成时间是：

$$time(i_j) = f(i_j) + t(i_j) = \sum_{k=1}^{j-1} t(i_k) + t(i_j) = \sum_{k=1}^{j} t(i_j)$$

f 的总完成时间是

$$time(f) = \sum_{k=1}^{n} time(i_k) = \sum_{j=1}^{n} t(i_j)(n-j+1)$$

由于平均完成时间为 $\frac{1}{n}time(f)$，当 $time(f)$ 取得最小值时得到最优解 f.

假如在 f 中存在逆序 (i_j,i_k) 使得 $j<k$ 但是 $i_j>i_k$，必有相邻的逆序. 交换具有逆序的相邻元素 i_j 与 i_k，得到解 g. 那么总的完成时间之差是

$$time(g) - time(f) = t(i_k) - t(i_j) \leqslant 0$$

因此 g 也是最优解. 至多经过 $n(n-1)/2$ 次交换，去除所有的逆序，就可以使得 f 转变成贪心法的解，从而证明了贪心法的解是最优解.

算法在最坏情况下的时间复杂度为 $O(n\log n)$.

4.13 15%（贪心策略 2%，算法证明 8%，算法表述 3%，时间复杂度 2%）

4.13 使用贪心法.

贪心策略：优先安排前 D 个罚款最多的作业.

算法的正确性证明使用交换论证的方法. 先给出以下命题.

命题 4.11 设作业调度 f 的安排次序是 $<i_1, i_2, \cdots, i_n>$，那么罚款

$$F(f) = \sum_{k=D+1}^{n} m(i_k)$$

证 显然最优调度没有空闲时间，不妨假设作业是连续安排的. 因为每项作业的加工时间都是 1，在截止时间 D 之前可以完成 D 项作业. 只有在 D 之后安排的 $n-D$ 项作业，即作业 $i_{D+1}, i_{D+2}, \cdots, i_n$ 是被罚款的作业.

从命题 4.11 可以直接推出以下结果：令 S 是 $n-D$ 项罚款最少的作业构成的集合.

(1) 对于 S 中的作业 i 和 j，或对于 $J-S$ 中的作业 i 和 j，交换 i 和 j 的加工顺序不影响总罚款.

(2) 对于作业 i 和 j，$m(i)<m(j)$，调度 f 将 i 安排在 D 之前，j 安排在 D 之后，那么交换作业 i 和 j，得到调度 g，则 g 的罚款将减少. 因为

$$F(g) - F(f) = m(i) - m(j) < 0$$

根据上述分析，不难看到，把罚款最小的 $n-D$ 项作业安排在最后将使得罚款总额达到最小. 设计以下算法：

1. 利用 Select 算法从 $m(1), m(2), \cdots, m(n)$ 中选第 $n-D$ 小，记作 m^*.
2. 用 m^* 与其他 $n-1$ 个 $m(i)$ 进行比较，找出比 m^* 小的 $n-D-1$ 个 $m(i)$.
3. 将上述 $n-D$ 项 $m(i)$（含 m^* 在内）对应的作业从 D 时刻开始以任意顺序安排加工.
4. 将剩下的 D 项作业以任意顺序安排在 $0, 1, \cdots, D-1$ 时刻加工.

上述算法本质上属于贪心法. 根据前面的分析，它的总罚款数与完全按照 $m(i)$ 从大到小安排作业的贪心法是一样的. 但是贪心法需要对作业按照 $m(i)$ 的大小排序，最坏情况下的时间复杂度是 $O(n\log n)$；采用上述基于 Select 选择过程的算法需要的时间不超过 $O(n)$.

4.18 25%（（1）贪心策略 2%，算法证明 8%，算法表述 3%，时间复杂度 2%；（2）算法 8%，时间复杂度 2%）

4.18 （1）按照标号从小到大考查广告. A 是被选取的广告标号集合，初始 $A=\{1\}$. 设 j 是当前 A 中最大标号. 考查第 k 个项目时，需检验 $s(k)\geqslant d(j)$. 若是，则把 k 加入 A；否则考查 $k+1$ 项.

命题 4.17 对任意正整数 k，算法选择了 k 项广告 $i_1=1,i_2,\cdots,i_k$，存在包含 $i_1=1,i_2,\cdots,i_k$ 的最优解.

证 对 k 归纳.

$k=1$，一定存在最优解包含 1. 如若不然，设最优解 A' 中广告的最小标号是 j，将 j 替换成 1，即

$$A^*=(A'-\{j\})\cup\{1\}$$

那么 A^* 是包含标号 1 的最优解.

假设对任意正整数 k 命题为真，即算法到第 k 步选择了 k 项广告 $i_1=1,i_2,\cdots,i_k$，且存在包含 $i_1=1,i_2,\cdots,i_k$ 的最优解 A，那么有

$$A=\{i_1,i_2,\cdots,i_k\}\cup B$$

令 S' 是 $S-\{i_1,i_2,\cdots,i_k\}$ 中开始时间不小于 $d(k)$（与已经选中的广告相容）的那些广告的集合. 那么不难证明 B 是 S' 的一个最优解. 根据归纳基础的证明可以知道，第 1 步选择标号最小的广告总能导致最优解，因此 S' 中有一个包含广告 i_{k+1} 的最优解 B^*. B^* 与 B 都是最优解，因此 $|B^*|=|B|$. 在 A 中用 B^* 替换 B 就得到如下最优解

$$A'=\{i_1,i_2,\cdots,i_k\}\cup B^*=\{i_1,i_2,\cdots,i_k,i_{k+1},\cdots\}$$

这恰好包含了算法前 $k+1$ 步的选择. 根据归纳法，命题得证.

算法在最坏情况下的时间复杂度为 $O(n)$.

（2）用动态规划算法，设 $F[k]$ 表示考虑前 k 个广告~~且选中第 k 个广告~~的最大效益. 定义 $p(k)$ 是与 k 相容（时间不重叠）且标号小于 k 的广告中的最大标号. 那么 $F[k]$ 满足如下递推关系：

$$F[k]=\max\{F[k-1],F[p(k)]+v(k)\},\quad k>1$$
$$p(k)=\max\{i\mid i\in\{1,2,\cdots,k-1\}\text{ 且 }d(i)\leqslant s(k)\}$$
$$F[1]=v(1)$$

用标记函数 $i[k]$ 记下当选到第 k 项广告时，紧接在它前边安排的广告标号，即

$$i[k]=\begin{cases}p(k), & \exists\, p(k)\in\{1,2,\cdots,k-1\}\text{ 且 }F[p(k)]+v(k)\geqslant F[k-1]\\ 0, & \text{否则}\end{cases},\quad k\geqslant 1$$

算法得到的最大效益是

$$\max\{F[k]\mid k=1,2,\cdots,n\}$$

算法输出数组 F 和 i 以及取得最大效益时的 k 值（最后一项广告的标号）.

为追踪解 A 需要考查标记函数 i. 具体方法是：将 k 加入解 A. 接着追踪 $i[k]$，如果 $i[k]=j$，那么 j 是在 k 前面被选中的广告标号. 把 j 加入 A，将 k 更新为 j，继续追踪新的 $i[k]$，直到 $i[k]=0$ 为止.

上述算法再最坏情况下的时间复杂度是$O(n^2)$

使用二分查找预先计算 p(k)，时间复杂度为 O(n log n)

4.20 15%（贪心策略 2%，算法证明 8%，算法表述 3%，时间复杂度 2%）

4.20 使用贪心法.

贪心策略：排序 r_i 为递减次序，使得 $r_1 \geqslant r_2 \geqslant \cdots \geqslant r_n$，依次购买.

命题 4.19 按照上述顺序购买软件许可证花费的总钱数最少.

证 设最优解为 OPT(I)，假设在 OPT(I) 存在逆序，即 $r_j < r_i$，但是 j 在 i 前面购买. 一定有相邻的逆序，即存在 i 和 j，使得 j 在 i 前面与 i 相邻. 设 j 是第 t 个月购买，i 是第 $t+1$ 个月购买. 交换 i 与 j 得到解 $S(I)$，那么花费之差

$$
\begin{aligned}
& V(\mathrm{OPT}(I))V(S(I)) \\
&= (r_i^{t+1} \times 1000 + r_j^t \times 1000) - (r_i^t \times 1000 + r_j^{t+1} \times 1000) \\
&= [r_i^t(r_i - 1) - r_j^t(r_j - 1)] \times 1000
\end{aligned}
$$

由于 $r_i \geqslant r_j$，于是 $r_i - 1 \geqslant r_j - 1$ 且 $r_i^t \geqslant r_j^t$，得到上式 $\geqslant 0$.

算法的时间复杂度为 $O(n\log n)$.