高等理工学院《算法设计与分析》
(2020 年秋季学期)

第三次作业参考答案

1 探险家分组问题 (20 分)

营地中共有 n 个探险家，第 i 个探险家的经验值为 $e_i(1 \leq i \leq n)$。现他们希望组成尽可能多的队伍前去探险。探险家组建队伍需满足如下规则：

1. 探险家可以不参加任何队伍，即留在营地；
2. 如果第 i 个探险家参加了某支队伍，那么该队伍的人数应不小于其经验值 e_i。

请设计一个尽可能高效的算法求出最多可组建几支队伍前去探险，并分析其时间复杂度。

例如有 $n = 5$ 名探险家，其经验值分别为 $e = \{2, 1, 2, 2, 6\}$，则可组建 $(1, 2)$,$(2, 2)$ 两支队伍，把经验值为 6 的探险家留在营地。

解：

1. 贪心策略

将所有探险家按照经验值 e_i 从小到大排序，遍历所有探险家：

1. 将探险家加入当前分组。
2. 如果当前探险家组内探险家人数与当前组内经验值最大的探险家相同，则将这些探险家分成一组，分组数量增加，当前分组重新置为空；

2. 策略证明

从策略上讲，每个组的大小与本组中经验值最大的探险家相同。(若人数更多，则可以把多余的人从本组移除加入其他组，或新生成一个组)

那么假设当前组内最大经验值为 e，则其可以让 $e-1$ 个经验值小于等于 e 的探险家不成为其他经验值最大的探险家。故我们可以选择权值最靠近 e 的 $e-1$ 个探险家。

换言之，假设贪心策略选取的 k 名经验值最大的探险家为 $p_1 \leq p_2 \leq \cdots \leq p_k$。而最优解为选择了 m 名经验值最大的探险家 $q_1 \leq q_2 \leq \cdots \leq q_m$ (这里 $m \geq k$)。

注意到 $p_t < q_t$，否则贪心算法会直接选取 q_t 来替代 p_t。而我们将经验值从小到大排序，这也就是说，我们可以用 p_t 选择的分组来替换 q_t 选择的分组（因为 p_t 对应经验值更小，需要的人数更少)。

则此时 $q_1, q_2, \cdots, q_{t-1}, p_t, q_{t+1}, \cdots, q_m$ 是贪心策略可以选出的一个可行解，这与 $k < m$ 矛盾，故贪心策略正确。

3. 时间复杂度分析

遍历整个数组的时间复杂度为 $O(n)$，而对所有元素排序的时间复杂度为 $O(n \log n)$，故总时间复杂度为 $T(n) = O(n \log n)$。

该算法的伪代码如 Algorithm 1 所示。

2 任务调度问题 (20 分)

给定 n 个任务，第 i 个任务用 (a_i, b_i) 表示 $(1 \leq i \leq n)$。其含义为：如果完成了第 i 个任务，则在其完成后的第 a_i 天，可以获得 b_i 的收益。

在接下来的 m 天中，每天可选择至多一个任务完成，且每个任务不能重复完成。

Algorithm 1 $group(n, e[1..n])$

Input:

长度为 n 的数列 e 表示探险家的经验值

Output:

最多分组数量

```
group ← 0
cnt ← 0
将 e 从小到大排序
for i : 1 → n do
    cnt ← cnt + 1
    if cnt = a[i] then
        group ← group + 1
        cnt ← 0
    end if
end for
return group
```

请设计一个尽可能高效的算法制定完成任务的计划，使得第 m 天结束时所获得的任务总收益最大。并分析其时间复杂度。

例如有 $n = 3$ 个任务，$m = 5$ 天。任务为 $\{(a_i, b_i)\} = \{(3, 4), (4, 5), (1, 2)\}$。则第 1 天可以完成任务 $(3, 4)$，可在第 $1 + 3 = 4$ 天获得 4 的收益。第 2 天可以完成任务 $(1, 2)$，可在第 $2 + 1 = 3$ 天获得 2 的收益。总收益为 6。

解：

1. 性质观察

注意到在第 i 天不能进行时长超过 $a = m - i$ 的任务。故我们可以从第 m 天开始倒序考虑每一天的工作。

2. 贪心策略

从第 m 天倒序开始观察，每天完成可以完成的任务中收益最大的一个。

3. 策略证明

如存在一个方案 $p_1, \cdots, p_m$ 比该方案 $q_1, \cdots, q_m$ 更优。则必有一天 i 存在 p_i 对应的任务比 q_i 优，那么这与贪心选取的是当天可以完成的收益最大的任务矛盾。

4. 贪心步骤

故先将所有任务按照 a_i 为关键字从小到大排序。从第 m 天开始倒序考虑，第 i 天将 $a_i \leq m-i$ 的所有工作加入大根堆 H(以 b_i 为关键字)，然后每一天从 H 中取出当前可以获得最大收益的任务即可。

5. 时间复杂度分析

对任务做排序的复杂度为 $O(n \log n)$，每一天维护堆所需要的复杂度为 $O(\log n)$，故总时间复杂度为 $T(n, m) = O((n + m) \log n)$。

该算法的伪代码如 Algorithm 2 所示。

Algorithm 2 $task(n, m, a[1..n], b[1..n])$

Input:

任务总时间 m 天，以及 n 个任务 $\{(a_i, b_i)\}$。

Output:

最大总收益

```
v[t] ← {b_i | a_i = m − t}
ans ← 0
for t : m → 1 do
    将 v[t] 内的所有元素加入大根堆 H
    取出大根堆 H 的堆顶 w，若堆为空 w 置 0
    ans ← ans + w
end for
return ans
```

3 数列重排问题 (20 分)

给定一个包含 n 个元素的数列 $\{a_1, \cdots, a_n\}$。对任意一个 $1..n$ 的排列 $\{p_1, \cdots, p_n\}$，其价值定义为 $\sum_{i=1}^{n-1} \max\{a_{p_i}, a_{p_{i+1}}\}$。

请设计一个尽可能高效的算法求出一个价值最小的排列并分析其时间复杂度。如果这样的排列有多个，请给出**字典序**最小的一个排列。

两个排列 $\{p_1, \cdots, p_n\}$ 和 $\{q_1, \cdots, q_n\}$ 的**字典序**采用如下规则比较：

1. $\forall 1 \le i \le n, p_i = q_i$，则两个数列的字典序相等。
2. $\exists 1 \le j \le n, p_j < q_j \wedge \forall 1 \le i < j, p_i = q_i$，则数列 p 的字典序较小。
3. $\exists 1 \le j \le n, p_j > q_j \wedge \forall 1 \le i < j, p_i = q_i$，则数列 q 的字典序较小。

例如 $\{a_1, a_2, a_3\} = \{4, 7, 5\}$，有 4 种价值最小的排列：$\{1, 3, 2\}, \{2, 1, 3\}, \{2, 3, 1\}, \{3, 1, 2\}$。其价值均为 $5 + 7 = 12$，其中字典序最小的排列为 $\{1, 3, 2\}$。

解：

1. 性质观察

一个排列的权值实质是由 $n - 1$ 个数相加而成，而每个数最多出现 2 次，最少出现 0 次。若一个数出现不到 2 次，则必有被比他大的数代替出现之。

故，最理想的情形是除了最小的数以外的每个数都出现了一次。一个可行的构造方案是：选择一个先递减再递增的序列。

2. 方案选择

注意到题目希望求得一个字典序最小的方案，故需要贪心选择一个字典序最小的递减序列。

3. 贪心策略

1. 递减序列: 从头开始遍历数组，先将第一个元素加入递减序列，每次考察递减序列的最末元素是否大于等于当前元素，如果是，则将当前元素也加入递减序列。
2. 递增序列: 将剩余元素按照权值为第一关键字，原始位置为第二关键字，排序为递增序列即可。

4. 时间复杂度分析

选取递减序列的时间复杂度为遍历整个数组 $O(n)$，而构造递增序列需要将剩余元素排序 $O(n \log n)$，故总时间复杂度为 $T(n) = O(n \log n)$

该算法的伪代码如 Algorithm 3 所示。

Algorithm 3 $order(n, a[1..n])$

Input:

长度为 n 的数列 a

Output:

重排的排列 p

```
p ← ∅, p' ← ∅
将 1 加入 p 末尾
lst ← a[1]
for i : 2 → n do
    if lst ≥ a[i] then
        将 i 加入 p 末尾
        lst ← a[i]
    else
        将 i' 加入 p' 末尾
    end if
end for
将 p' 按照其元素对应的 a[i] 为第一关键字，i 为第二关键字排序
return p + p' (将 p' 拼接在 p 之后)
```

4 交通建设问题 (20 分)

现有 n 个城市，初始时任意两个城市之间均不可互达。现有两种交通建设方案：

1. 花费 $c_{i,j}$ 的代价，在城市 i 和城市 j 之间建设一条道路，可使这两个城市互相可达。
2. 花费 a_i 的代价，在城市 i 建设一个机场，可以使得其与其他所有建设了机场的城市互相可达。

只要两个城市 i 和 j 之间存在一条可达的路径，则这两个城市也互相可达。请设计一个尽可能高效的算法求出使所有城市之间互相可达所需的最小花费，并分析其时间复杂度。

解：

1. 问题思路

如果问题不存在建立机场，则为一个简单的最小生成树问题（$G(V,E)$ 其中 V 是 n 个城市,E 是每个城市之间建设道路的费用构成的集合），对于建立机场，相当于可以虚拟一个新点 sky，增加每个点 i 到 sky 存在一条 a_i 权值的边，为一个新图 G_{sky}。

2. 问题求解

即仅需要对原图 G 求一个最小生成树 T 和对新图 G_{sky} 求一个最小生成树 T_{sky}。比较两个树的大小，选较小的那个为最少花费即可。

3. 时间复杂度分析

故本题仅需要求解两次最小生成树即可，注意到图是稠密图，采用 Prim 算法求解最小生成树的时间复杂度为 $O(n^2)$。故总时间复杂度为 $T(n)=O(n^2)$ 为所求。

该算法的伪代码如 Algorithm 4 所示。

Algorithm 4 $city(n, a[1..n], c[1..n][1..n])$

Input:

$n \times n$ 的矩阵 c，表示任意两点之间道路的费用，长度为 n 的数列 a，表示建设机场的代价

Output:

使得所有城市可以互相到达的最小花费

$V \leftarrow \{1, \cdots, n\}$
$E \leftarrow \{< u, v, w > | u, v \in V, w = c_{u,v0}\}$
$V' \leftarrow V \cup \{sky\}$
$E' \leftarrow E \cup \{< u, sky, w > | u \in V, w = a_u\}$
$T \leftarrow$ PrimMST(V,E)
$T' \leftarrow$ PrimMST(V',E')
if T 的边权和小于 T' **then**
  return T 的边权和,T 两个点之间的边表示建设道路
else
  return T' 的边权和,T' 其中连到 sky 的边表示建设机场
end if

5 迷宫逃离问题 (20 分)

给定一个 $m \times n$ 的迷宫，其入口和出口分别为 $(1,1)$ 和 (m,n)。每个格子有两种状态：

1. $c_{i,j}=0$，表示这个格子是空格子，可以通过；
2. $c_{i,j}=1$，表示这个格子是障碍物，不可通过。

入口 $(1,1)$ 和出口 (m,n) 均为空格子。在迷宫中可从某个格子 (i,j) 移动到与其相邻的空格子 $((i,j-1),(i,j+1),(i-1,j),(i+1,j)$ 其中之一)，消耗体力为 1。

现可将至多 1 个障碍物移除，使其对应的格子的变为空格子。请在此基础上设计一个尽可能高效的算法，求出从入口 $(1,1)$ 到出口 (m,n) 需消耗的最小体力，并分析其时间复杂度。

解：

1. 搜索状态

注意到经过每个格子时有尚未移除障碍物，和已经移除障碍物两种状态，故状态空间是 $n \times m \times 2$ 的。可以尝试使用广度优先搜索。考虑用状态 $(x, y, 0)$ 表示在 (x, y) 点还没有移除障碍物，状态 $(x, y, 1)$ 表示在 (x, y) 点已经移除了某一个障碍物。

2. 搜索顺序

在状态 $(x, y, 0)$ 可以扩展到如下几个状态:

1. $(x, y-1, 0), (x-1, y, 0), (x, y+1, 0), (x+1, y, 0)$ 如果对应的新的格子是可以通过的空格子;

2. $(x, y-1, 1), (x-1, y, 1), (x, y+1, 1), (x+1, y, 1)$ 如果对应的新的格子是障碍物，则移除之。

而状态 $(x, y, 1)$ 只能扩展到四个方向相邻的空格子中去:$(x, y-1, 1), (x-1, y, 1), (x, y+1, 1), (x+1, y, 1)$。

3. 时间复杂度分析

注意到每个状态至多只会被搜索一次，每次扩展时间是常数级别的，故总的时间复杂度为 $T(n, m) = O(n \times m)$。

该算法的伪代码如 Algorithm 5 所示。

Algorithm 5 $maze(m, n, c[1..m][1..n])$

Input:

给定的迷宫 $c_{m \times n}$

Output:

最小消耗的体力

```
Q ← ∅
vis[1..m][1..n][0..1] ← 0
将 (1,1,0) 状态加入队列 Q
vis[1][1][0] ← 1
eng[1][1][0] ← 0
while Q 非空 do
    从 Q 队首取出状态 (x, y, w)
    for (x', y') ← {(x-1, y), (x+1, y), (x, y-1), (x, y+1)} do
        if c_{x',y'} == 0 then
            if vis[x'][y'][w] = 0 then
                vis[x'][y'][w] = 1
                eng[x'][y'][w] ← eng[x][y][w] + 1
                将状态 (x', y', w) 加入队列 Q
            end if
        else
            if w == 0 ∩ vis[x'][y'][1] = 0 then
                vis[x'][y'][1] = 1
                eng[x'][y'][1] ← eng[x][y][w] + 1
                将状态 (x', y', 1) 加入队列 Q
            end if
        end if
    end for
end while
return min(eng[m][n][0], eng[m][n][1])
```