计算机算法设计与分析

第 3 次作业

刘炼
202128013229021

Problem 1

Assignment Problem2

Solution

根据题目的描述，任务实际上分成两步完成。在 supercomputer 上计算的时间是排队的，在后续的 PC 处理上是可以并行的。故执行 n 个任务，在 supercomputer 中的时间是固定的，均为 $\sum_{i=1}^{n} p_i$，而最终结束的时间取决于最晚的任务 j 结束的时间 $\sum_{i=1}^{j} p_i + f_j$。故思路为，将 PC 执行时间长的放在前面，而将 PC 执行时间短的放在后面执行。

Pseudo-Code shown as follows.

Algorithm 1 Optimal Schedule

Input: P, F ▷ P and F respectively denote time consumed on supercomputer and PC.

```
Initialization: initialize an array T
Use QuickSort to sort F in a reversed order and save indexs.
T ← indexs of sorted F
```

Output: Schedule T

Greedy-Choice Property

根据算法的分析，则假设调度后的任务序列满足如下条件：$f_i \geq f_j$ 当且仅当 $i \leq j$，假设 $i \neq j$。设调度前 j 个任务的时间为 $T(j)$, 假设这样的调度不是最优调度，那么存在一个调度，其中交换 i 和 j，那么此时对于任务 i，执行时间为 $\sum_{k=1}^{j} p_k + f_i$，故执行前 j 个任务的时间 $T'(j) \geq T(j)$，这与之前不是最优调度的假设不符。故实际上，这样一种 greedy 调度满足最优调度。

根据上面的分析，最优子结构应该具有这样的性质，假设任务集合为 S, 最后一个任务的执行时间（假设最后一个任务为 k），应该为 $\sum_{i=1}^{j} pi + f_k$。故应该描述为

$$T(n) = \min\{\max(T(S-k), \sum_{i=1}^{n} p_i + f_k)\},\ k \in [1, n] \tag{1}$$

Proof of the correctness

使用结构归纳法证明如下：

- Initialization：对于数量为 1 的任务，调度需要的总时间为 $T(1) = p_1 + f_1$。任务调度满足降序有序性。
- Generalization: 假设对于数量为 $n-1$ 的任务，调度顺序为 $1, 2, ..., n-1$，其执行的总时间为 $T(n-1)$，任务满足 PC 上执行时间的降序有序性。故对于数量为 n 的任务，根据两种情况考虑：
  （1）任务 n 中 PC 执行时间 f_n 的执行时间要少于前 $n-1$ 个任务，此时执行任务 n 的时间为 $\sum_{i=1}^{n} p_i + f_n$，如果交换该任务与此前 $n-1$ 中任意一个任务，那么该时间增加，最终的 $T(n)$ 增加，故不满足条件，故应保持，此时同样满足任务 n 的降序有序性。
  (2) 任务 n 中 PC 执行时间 f_n 的执行时间要大于从 i 到 $n-1$ 个任务，如果还保持原有的顺序进行调度，那么存在一种调度，即将任务 n 安排在任务 i 之前，此时 $T'(n) < T(n)$，故应该进行这样的调度，进行调整后，调度满足任务 n 在 PC 执行时间上的降序有序性。

得证

Problem 1 continued on next page...

Analysis of Complexity

时间复杂度: 根据分析，实际上需要对 PC 上的任务执行时间进行一次排序，不妨采用快速排序方法，则时间复杂度为 $O(nlogn)$

空间复杂度: 采用快速排序方法，需要一个数组来存储调度后的顺序，故空间复杂度为 $O(n)$

Problem 2

Assignment Problem3

Solution

根据题目的描述，实际上就是尽可能地，让两个人坐一条船。而考虑要使尽量多的两个人做一条船，那么实际上，对于最轻的那些人，就可以尽可能找一个刚好满足需要的人，与其拼船坐。故思路为，将人群按照体重从轻到重进行排序，然后从头挑出还没坐船的最轻的人，并从队列中，倒序找到第一个与其匹配，能够拼船坐的人。

Pseudo-Code shown as follows.

Algorithm 2 Minimum Boats

Input: W, T ▷ P denotes weights of different people, and T is the boat's weight limit.

Initialization: initialize number of boats $x \leftarrow 0$
Use QuickSort to sort W in ascending order.
$i \leftarrow 0, j \leftarrow n-1$ ▷ n is the number of people
while i < j **do**
    if $W[i] + W[j] \leq T$ **then**
        $i += 1$
        $x += 1$
    end if
    $j -= 1$
end while
$x \leftarrow (n-x)$

Output: Number x

Greedy-Choice Property

根据算法的分析，在按照升序排列后的体重数组为 W，满足：$W[i] \leq W[j]$ 当且仅当 $i \leq j$。假设该分配不是最优分配，存在一个分配使得使用的船的数目更少，即满足能够将两条船上的两个人合并起来或者三条船上的四个人合并成两条船上的人。根据前面的分析，显然两条船上的两个人合并起来是不可行的。下面分析，对于三条船 s, u, v，其中 s 船中有两个人，体重分别为 $s_1, s_2, s_1 < s_2$，船 u, v 中，均有一个人，体重为 u_1, v_1。不妨假设 $u_1 < v_1$，如果 $v_1 > s_2$，由于前面分析，s_2 是最大的能与 s_1 搭配的点，此时显然不能满足将 s_1 和 v_1 进行拼船。如果有 $s_1 < u_1 < v_1 < s_2$，根据分析，$u_1 + v_1 > T$，故此时调换后亦不能用两艘船进行拼接，故不存在一个更优调度策略。

根据上面的分析，最优子结构应该具有这样的性质，假设原来的人群集合为 P, 那么其可以从中选择两个人去拼船或者坐两条船。假设选择的人为 u, v，最优子结构可以描述为如下

$$T(P) = \min \begin{cases} T(P - \{u, v\}) + 1, W[u] + W[v] \leq T \\ T(P - \{u, v\}) + 2, W[u] + W[v] > T \end{cases} \quad u, v \in P \tag{2}$$

Problem 2 continued on next page...

Proof of the correctness

使用结构归纳法证明如下：

- Initialization：对于集合 P, 若长度为 1 的人群，需要的船数为 1。在安排时，人群体重符合升序有序性。
- Generalization: 对于集合 P，在安排时，人群体重符合升序有序性，船的数量为 $T(p)$，当集合为 $P+u$，u 代表一个新加入的人，那么如果不考虑原来的有序性，则需要新加入一条船，船的数目变为 $T(p)+1$。可以将原集合 P 的顺序在有序加入 u 后变为 $p_1, p_2, ..., p_i, u, p_j, ..., p_n$, 对于上述结果，分两种情况考虑：
（1）假设 p_i 满足可拼船的较轻的要求，那么对于 u 而言，其也可能满足拼船的要求，并能从中找到一个人进行拼船，此时，船的数目可以减少为 $T(P)$
(2) 假设 p_i 不满足可拼船的较轻的要求，那么无法增加双人坐船的数目，此时船的数目为 $T(p)+1$。
总体而言，在两种情况下，将 u 插入并保持有序，都不会导致船数较不插入增加，反而有可能减少，故应始终保持 P 的升序有序性。

得证

Analysis of Complexity

时间复杂度: 根据分析，实际上需要对人群的体重进行一次排序，不妨采用快速排序方法，则时间复杂度为 $O(nlogn)$，后序只需要一次遍历过程，时间复杂度为 $O(n)$，故总体时间为 $O(nlogn)$
空间复杂度: 采用快速排序方法，排序和遍历操作都是本地操作，不需要额外空间，故空间复杂度为 $O(1)$

Problem 3

Assignment Problem5

Solution

根据题目的描述，要满足所有的 building 不降序排列，且每次可以给连续一段进行加一操作。考虑这样的情况，如果只加高中间一段，而不对后面一段加高，那么如果中间的高度超过了后面，在后续操作中，需要多进行操作来加高后面的 building，故总体上连续一段加高，要从该加高的位置一直加到最后一个 building。

Pseudo-Code shown as follows.

Algorithm 3 Minimum Operations

Input: P ▷ A and F denotes height for buildings.

```
Initialization: initialize x ← 0
for i = 2; i ≤ n; i++ do
    x += max(0, (A[i − 1] − A[i]))
end for
```

Output: Operation Number x

Greedy-Choice Property

根据之前的分析，实际上每一次操作应该对从左到右的第一个降序进行操作，即假设 $a_i > a_{i+1}$, 那么应该对从 $i+1$ 到 n 的每一个 building 都进行添加高度操作。假设这样的添加方式不是最优方式，那么至少存

Problem 3 continued on next page...

在一次操作，使得只从 $i+1$ 到 $j, j < n$ 处进行了添加，有两种情况进行分析：
(1) 如果 $a_j \geq a_{j+1}$，此时由于 j 处进行了添加操作，故要满足非降序情况，则要在 $j+1$ 出进行 $a_j - a_{j+1} + 1$ 次操作，而原来只需要 $a_j - a_{j+1}$ 次操作，故此时不满足最优。
(2) 如果 $a_j < a_{j+1}$，此时由于 j 处进行了添加操作，仍然满足非降序情况，这和在后续添加操作是一样的次数，故同时满足最优。故综上两种情况分析，一种 greedy 的将从第一个不满足非降序点到结尾出的所有 buildings 都添加的策略是一种最优策略。

根据上面的分析，最优子结构应该具有这样的性质，即对于 n 个 buildings 的最少添加操作次数，应该为后 $n-1$buildings 进行操作的数目和保持第一个和第二个 buildings 的非降序性需要操作的数目之和。故应该描述为

$$T(n) = T(n-1) + max(0, A[1] - A[2]) \tag{3}$$

Proof of the correctness

使用结构归纳法证明如下：

- Initialization：对于数量为 1 的 buildings，需要的操作次数为 0。
- Generalization: 假设对于数量为 $n-1$ 的 buildings，需要的操作次数为 $T(n-1)$，则对于数量为 n 的 buildings，假设将第 n 个 building 放在最左端，分两种情况讨论：
  （1）如果该 building 小于等于原来最左端的 building 的高度，那么不需要进行操作，只需要对后续 $n-1$ 个 buildings 进行操作
  (2) 如果该 building 大语原来最左端的 building 的高度，那么将后面 $n-1$ 的 building 的高度同时提升 k 次，其中 k 为高度差，后满足该情况。故综上，与分析中的等式3相一致。

得证

Analysis of Complexity

时间复杂度: 根据分析，只需要进行一次循环操作，故时间复杂度为 $O(n)$
空间复杂度: 根据算法所示，不需要额外的数组空间，故空间复杂度为 $O(1)$