计算机学院《算法设计与分析》
(2023 年秋季学期)

第三次作业参考答案

1 分蛋糕问题 (20 分)

给定 n 块体积不同的蛋糕，其体积分别用 $a_1, \cdots, a_n$ 表示。现要从中挑选出 $k(k < n)$ 块蛋糕分给同学们。不妨记选出的蛋糕的编号为 $s_1, \cdots, s_k(1 \leq s_1 < \cdots < s_k \leq n)$，则这次分配的不公平度为

$$\max\{a_{s_1}, \cdots, a_{s_k}\} - \min\{a_{s_1}, \cdots, a_{s_k}\}$$

请设计一个尽可能高效的算法制定蛋糕的选取方案，使得选出蛋糕的不公平度最小，请描述算法的核心思想，**必要时给出证明**，给出算法伪代码并分析其对应的时间复杂度。

解:

1. 贪心策略

将蛋糕按照体积从小到大排序，$\min_{i=1}^{n-k+1}\{a_{i+k-1} - a_i\}$ 为最小不公平度。此时记 $p = \arg\min_{i=1}^{n-k+1}\{a_{i+k-1} - a_i\}$。则选取排序后的 $a_p \sim a_{p+k-1}$ 块为使得不公平最小的一个选取方案。

2. 策略证明

考虑以 a_i 为最小元素的集合 S_i，希望最小化 S_i 的不公平度即为最小化 S_i 集合的最大元素。此情况即为选择了大于等于 a_i 的最小的前 k 个元素。

故上述贪心策略实际上考察了每一个元素作为最小元素的所有集合，故策略正确。

3. 时间复杂度分析

注意到仅仅需要做一次排序 $O(n \log n)$ 和将整个数组扫描一趟 $O(n)$。故总的复杂度为 $T(n) = O(n \log n)$。

参考伪代码如1所示。

Algorithm 1 $alloc(n, k, a[1..n])$

Input: n 块蛋糕 $a[1..n]$，及所选的块数 k
Output: 不公平度最小的方案

```
将 a[1..n] 从小到大排序
minval ← ∞
for i : 1 → n − k + 1 do
    if a_{i+k−1} − a_i ≤ minval then
        minval ← a_{i+k−1} − a_i
        p ← i
    end if
end for
return minval, a[p..p + k − 1]
```

---

2 分糖果问题 (20 分)

F 同学手中有 n 袋糖果，其中第 i 袋中有 a_i 颗糖。F 同学希望在**不拆开糖果袋**的前提下把一些糖果分给他的妹妹，但是他希望分完之后留给自己的糖果总数要**大于**妹妹得到的糖果总数。

请设计一个算法帮 F 同学计算他最少要给自己留多少袋糖果才能满足要求，请描述算法的核心思想，**必要时给出证明**，给出算法伪代码并分析其对应的时间复杂度。

例如，F 同学持有 3 袋糖果，分别装有 $\{2,1,2\}$ 颗糖果，这时他至少需要给自己留 2 袋糖果，这两袋糖果的数量可以是 $\{1,2\}$，或者是 $\{2,2\}$。

解：

1. 贪心策略

将 n 袋糖果按照糖果数量从大到小排序，然后依次选择糖果，直到所选的糖果总数大于剩下的糖果总数。此时所选取的糖果袋数即为满足题意的最少袋数。

2. 策略证明

上述贪心策略的思路相对直观：若要选取糖果的袋数最少，应该选取糖果数量最多的那几袋糖果。可以使用替代法对上述策略的正确性进行证明（类似部分背包问题）。假设有贪心解 S 与最优解 S^*，并对这两个解所选取的若干袋糖果 s_i、s_i^* 按照糖果数量从大到小排序，并依次进行对比。若 s_i 与 s_i^* 不同，由于贪心策略选取的是当前数量最多的一袋糖果，故有 $a_{s_i} \geq a_{s_i^*}$，进而将最优解中的 s_i^* 替换为 s_i 后仍能满足题意且袋数不变。以此类推，可以将最优解替换为贪心解，证明了策略的正确性。

3. 时间复杂度分析

排序的时间复杂度为 $O(n\log n)$，遍历依次选取糖果的时间复杂度为 $O(n)$。故总的复杂度为 $T(n) = O(n\log n)$。

伪代码见 Algorithm2。

Algorithm 2 $candy(n, a[1..n])$

Input: 糖果总袋数 n，每袋糖果的数量 $a[1..n]$。

Output: 满足要求的最少糖果袋数。

$total \leftarrow 0$
for $i : 1 \rightarrow n$ **do**
    $total \leftarrow total + a[i]$
end for
将 $a[1..n]$ 从大到小排序
$sum \leftarrow 0$
for $i : 1 \rightarrow n$ **do**
    $sum \leftarrow sum + a[i]$
    if $sum > total - sum$ **then**
        break
    end if
end for
return i

3 最大收益问题 (20 分)

某公司有一台机器，在每天结束时，该机器产出的收益为 X_1 元。在每天开始时，若当前剩余资金大于等于 U 元，则可以支付 U 元来升级该机器（每天最多只能升级一次）。从升级之日起，该机器每天可以多产出 X_2 元的收益。即是说，在执行 K 次升级之后，这台机器每天的产出为 $X_1 + K \times X_2$ 元。

该公司初始资金为 C 元，请你设计算法求出 n 天之后该公司拥有的总资金的最大值。请描述算法的核心思想，**必要时给出证明**，给出算法伪代码并分析其对应的时间复杂度。

解：

1. 贪心策略

根据机器产出的公式可以看出，每次升级其实是相互独立的。我们可以将每次升级看作花费 U 元购买了一台新机器，这台新机器每天可以产出 X_2 元。

2. 策略证明

因此，在第 i 天升级（购买新机器），这台新机器到第 n 天的总产出为 $(n-i+1)\times X_2$ 元。只要该总产出大于升级的成本 U，我们就应该在第 i 天进行升级。当然，还需要根据第 i 天时的剩余资金来判断这一天是否可以进行升级。

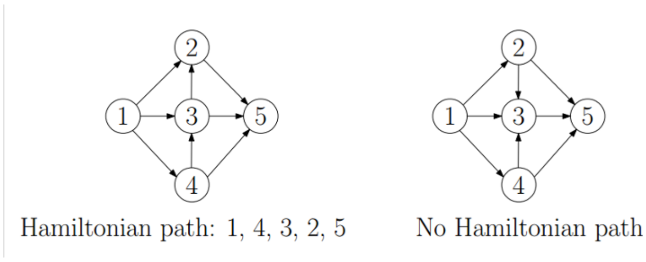

图 1: 哈密顿路径

算法的伪代码请参考 Algorithm 3。

3. 时间复杂度分析该算法需模拟每天的收益情况，时间复杂度为 $O(n)$。

Algorithm 3 $profit(X_1, X_2, U, C, n)$

Input: 机器初始产出 X_1，升级的收益 X_2，升级所需成本 U，初始资金 C 以及总天数 n

Output: 该公司 n 天后总资金的最大值

for $i : 1 \to n$ **do**
    if $(n - i + 1) \times X_2 > U$ **and** $C \geq U$ **then**
        $C \leftarrow C - U$
        $X_1 \leftarrow X_1 + X_2$
    end if
    $C \leftarrow C + X_1$
end for
return C

---

4 哈密顿路径问题 (20 分)

哈密顿路径 (Hamiltonian path) 是指图中每个节点都仅经过一次且必须经过一次的路径。对于一般的图结构来说，求解哈密顿路径的问题是 NP 难问题。然而，在有向无环图上寻找哈密顿路径的问题是存在多项式时间的解法的。

如图1所示，左侧图包含一条哈密顿路径 $1 \to 4 \to 3 \to 2 \to 5$，右侧图则不包含哈密顿路径。

给定一个有向无环图 $G = (V, E)$，请设计一个高效算法来寻找图 G 的一条哈密顿路径，如不存在哈密顿路径则返回 -1，请描述算法的核心思想，给出算法伪代码并分析其对应的时间复杂度。

解：

1. 问题分析

由于该图为有向无环图（DAG），因此图中最长路径如果长度为 $|V|$ 个节点，则该路径为哈密顿路径，否则不存在哈密顿路径，因此考虑在有向无环图 G 中进行动态规划，计算哈密顿路径等价于计算最长路径。

2. 问题求解

对每个点 u，记 $dp[u]$ 表示以点 u 结束的最长路径的长度。

则可写出递归式：$dp[u] = \max_{v,(v,u)\in E} dp[v] + 1$

初始状态：$dp[u] = 0$

为了保证 $dp[u]$ 可以正确更新，需要对图 G 进行拓扑排序，之后按照拓扑序的顺序计算 $dp[u]$ 的值即可。

为了输出哈密顿路径，需要使用数组 $b[u]$ 来记录以点 u 结束的最长路中 u 的前驱节点。

伪代码见算法5。

3. 时间复杂度分析时间复杂度为 $O(|V| + |E|)$。

Algorithm 4 $topo\_sort(G)$

Input: 图 G
Output: 顶点拓扑序

初始化空队列 Q
for $v \in V$ **do**
  if $v.in\_degree = 0$ **then**
    $Q.Enqueue(v)$
  end if
end for
定义数组 ans
while $not\ Q.is\_empty()$ **do**
  $u \leftarrow Q.Dequeue()$
  在数组 ans 后追加 u
  for $v \in G.Adj(u)$ **do**
    $v.in\_degree \leftarrow v.in\_degree - 1$
    if $v.in\_degree = 0$ **then**
      $Q.Enqueue(v)$
    end if
  end for
end while
return ans

Algorithm 5 $hamilton(n, G(V, E))$

Input: 节点数量 n, 图 G
Output: 哈密顿路径 $path$

拓扑排序结果 $a \leftarrow topo\_sort(G)$
最长路径长度 $dp[i] \leftarrow 0$
记录前驱节点 $b[i] \leftarrow -1$
$ans \leftarrow 0$
$pos \leftarrow -1$
for $i : 1 \rightarrow n$ **do**
  $pre[a_i] \leftarrow \{u | \langle u, a_i \rangle \in E, u \in V\}$
  for $j : 1 \rightarrow |pre[a_i]|$ **do**
    if $dp[a_i] < dp[pre[a_i][j]] + 1$ **then**
      $b[a_i] \leftarrow pre[a_i][j]$
      $dp[a_i] \leftarrow dp[pre[a_i][j]] + 1$
    end if
  end for
  if $ans < dp[a_i]$ **then**
    $ans \leftarrow dp[a_i]$
    $pos \leftarrow a_i$
  end if
end for
if $ans + 1 < n$ **then**
  return -1
end if
哈密顿路径 $path \leftarrow []$
while $pos \neq -1$ **do**
  $path.append(pos)$
  $pos \leftarrow b[pos]$
end while
return $path$

5 马的遍历问题 (20 分)

给定一个 $n{\times}m$ 的中国象棋棋盘，规定其左下角的格点为坐标原点，坐标系 x 轴和 y 轴分别沿右方以及上方延伸。现在点 (x, y) 上有一个马，该棋子与中国象棋的马移动规则相同，并且不允许移动到棋盘范围外。

请设计一个高效算法，计算一个距离矩阵 $D[n][m]$，$D[i][j]$ 表示马从 (x, y) 移动到 (i, j) 最少需要几步，请描述算法的核心思想，给出算法伪代码并分析其对应的时间复杂度。

解：

1. 搜索状态

搜索状态即为每个点的坐标，共包含 $n \times m$ 个状态。

2. 搜索顺序

根据马的移动规则，可以分八种情况讨论：

$((2, 1), (1, 2), (-1, 2), (-2, 1), (-2, -1), (-1, -2), (1, -2), (2, -1))$

将初始点 (x, y) 作为广度优先搜索（BFS）起始点，根据以上移动规则进行搜索。

3. 时间复杂度分析

根据广度优先搜索的特点，每个状态第一次遍历到即为最优答案，因此每个搜索状态只会经过一次，故时间复杂度为 $O(n \times m)$。

伪代码见 Algorithm6。

Algorithm 6 $Travel(n, m, x, y)$

Input: 棋盘大小 n, m，起始位置 x, y。

Output: 最终答案 ans。

定义队列 Q

定义到达矩阵 $arrive[n][m] = -1$

距离矩阵 $D[n][m] = 0$

$pos[8] \leftarrow \{(2, 1), (1, 2), (-1, 2), (-2, 1), (-2, -1), (-1, -2), (1, -2), (2, -1)\}$

$Q.push(x, y)$

while $not\ Q.empty()$ **do**

    $x \leftarrow Q.front().first$

    $y \leftarrow Q.front().second$

    $Q.pop()$

    for $k : 1 \rightarrow 8$ **do**

        $x_{new} \leftarrow x + pos[k].first$

        $y_{new} \leftarrow y + pos[k].second$

        if $x_{new} < 1$ **or** $x_{new} > n$ **or** $y_{new} < 1$ **or** $y_{new} > m$ **then**

            continue

        end if

        if $arrive[x_{new}][y_{new}] == -1$ **then**

            $arrive[x_{new}][y_{new}] \leftarrow 0$

            $D[x_{new}][y_{new}] \leftarrow D[x][y] + 1$

            $Q.push(x_{new}, y_{new})$

        else

            if $D[x_{new}][y_{new}] > D[x][y] + 1$ **then**

                $D[x_{new}][y_{new}] \leftarrow D[x][y] + 1$

            end if

        end if

    end for

end while

return ans
