高等理工学院《算法设计与分析》
(2021 年秋季学期)

第三次作业参考答案

1 最大最小值问题 (20 分)

给定一个长度为 n 的正整数数组 $A = (a_1, a_2, \cdots, a_n)$，请设计一个高效算法找到一个闭区间 $[l, r](1 \leq l < r \leq n)$，使得 $\min(a_l, a_{l+1}, \cdots, a_r) \times \max(a_l, a_{l+1}, \cdots, a_r)$ 最大，**写出该算法伪代码**并分析时间复杂度。

解：

1. 贪心策略

考虑区间 $[l, r]$：

1. 若区间最大值为 a_l，考虑区间 $[l, r-1]$，最大值不变，最小值不会变小。
2. 若区间最大值不为 a_l，考虑区间 $[l+1, r]$，最大值不变，最小值不会变小。

对于区间 $[l, r](r - l + 1 > 2)$，一定存在子区间 $[l', r'](l \leq l' < r' \leq r, r' - l' < r - l)$ 的答案更优，因此最优答案的区间长度只可能为 2，因此遍历所有的长度为 2 的区间即可得到答案，答案为 $\max_{1 \leq i < n}(a_i \times a_{i+1})$。

2. 时间复杂度分析

贪心策略仅需遍历数组，因此时间复杂度为 $O(n)$。

伪代码见 Algorithm1。

Algorithm 1 $MinMax(a)$

Input:

数组 $a[1..n]$。

Output:

区间左右端点 l，r。

```
l ← 1
r ← 2
ans ← 0
for i : 1 → n − 1 do
    if ans < a[i] ∗ a[i + 1] then
        ans ← a[i] ∗ a[i + 1]
        l ← i
        r ← i + 1
    end if
end for
return l, r
```

2 道路铺设问题 (20 分)

给定一个长度为 n 米的道路，距离起点第 i 米处的高度为 d_i，每天可以选取一段区间 $[l, r]$，使得距离起点 l 米到 r 米的道路高度增加 1，即 $d_i = d_i + 1(l \leq i \leq r)$。

请设计一个高效算法，计算最少需要几天才能使道路高度全部一致，即满足 $d_i = d_j, \forall i, j, 1 \leq i < j \leq n$，**写出该算法伪代码**并分析时间复杂度。

解：

1. 贪心策略

道路高度最大值为 $m = \max_{1 \le i \le n} d_i$，因为只能提升道路高度，最优解会将全部道路高度提升到 m。

考虑两段相邻的道路 d_i 和 d_{i+1}，满足 $d_i \le d_{i+1} < m$，若将 d_{i+1} 提升高度 $m - d_{i+1}$，根据贪心思想，d_i 的高度也会提升 $m - d_{i+1}$。

考虑从左向右提升提升高度，距离起点 j 米的道路高度为 $d_j (d_j < d_{j-1})$，若 $d_{j+1} < d_{j-1}$，则可以“顺带”提升高度，因此找到 j 右侧第一个道路高度大于等于 d_{j-1} 的位置 k，将 $[j, k-1]$ 的道路高度提升至 d_{j-1}。

令道路高度 $d_0 = d_{n+1} = m$，则答案为 $ans = \sum_{i=0}^{n} \max(0, d_i - d_{i+1})$。

2. 时间复杂度分析

计算答案 ans 仅需遍历一遍数组，时间复杂度为 $O(n)$。

伪代码见 Algorithm2。

Algorithm 2 $Road(d)$

Input:

数组 $d[1..n]$。

Output:

区间左右端点 l，r。

$ans \leftarrow 0$
$m \leftarrow \max_i d[i]$
$d[0] \leftarrow m$
$d[n+1] \leftarrow m$
for $i : 0 \rightarrow n$ **do**
    $ans \leftarrow ans + max(0, d[i] - d[i+1])$
end for
return ans

---

3 数对链问题 (20 分)

给定 n 个数对 $P = ((a_1, b_1), (a_2, b_2), \cdots, (a_n, b_n))$。在每一个数对中，第一个数字总是比第二个数字小，即 $a_i < b_i$。

定义数对链为若干数对组成的链，每个数对的第二个元素小于下一个数对的第一个元素，例如，对于长度为 n 的数对链 $[(a_1, b_1), \cdots, (a_n, b_n)]$，需要满足 $b_i < a_{i+1} (1 \ge i < n)$。

请设计一个高效算法，选择 P 中的一些数对组成数对链，使得数对链长度最长，**写出该算法伪代码**并分析时间复杂度。

解：

1. 贪心策略

考虑已有的数对链的最后两个数对为 $(a_i, b_i), (a_j, b_j)$，很明显接下来的一个数对仅与 b_j 有关，若存在一个数对 (a_k, b_k) 满足 $b_i < a_k, b_k < b_j$，使用 (a_k, b_k) 替换 (a_j, b_j) 会使答案更优，因此可以通过对数对第二个元素排序，优先取第二个元素更小的数对。

2. 时间复杂度分析

数组排序时间复杂度为 $O(n \log n)$，遍历贪心时间复杂度 $O(n)$，总的时间复杂度为 $O(n \log n)$。

伪代码见 Algorithm3。

Algorithm 3 $PairChain(d)$

Input:

数组 $d[1..n]$。

Output:

区间左右端点 l，r。

```
sort(P) //根据数对第二个元素从小到大排序
minb ← 0
ans ← []
for i : 1 → n do
    if P[i].first > minb then
        minb ← P[i].second
        ans.append(P[i])
    end if
end for
return ans
```

4 删数问题 (20 分)

给定一个用字符串表示的正整数 $n(1 \le n \le 10^{250})$（不包含前导 0），去掉其中任意 k 个数字后，剩下的数字按原左右次序将组成一个新的非负整数（允许包含前导 0）。

请设计一个高效算法，找到一种删数方案使得剩下的数字组成的新数字最小，**写出该算法伪代码**并分析时间复杂度。

解：

1. 贪心策略

数字长度 n 和删除位数 k 固定，则新数字长度固定，因此我们优先最小化最高位数字。

从原数字高位开始考虑，假设原来第一位是 a，若第 $2 \sim k+1$ 位的最小值 b 小于 a，则删去第一个 b 前的所有数字，否则保留第一位数字。

更新数字 n 和 k，考虑下一位数字，重复以上过程，直至数字最后一位，若剩余位数小于等于当前 k，则直接删除剩余数字。

2. 时间复杂度分析

每次需要遍历前 k 位，因此复杂度为 $O(len(n) \times k)$。

伪代码见 Algorithm4。

Algorithm 4 $Delete(d)$

Input:

字符串 n。

Output:

最终答案 ans。

定义 ans 为空字符串
$i \leftarrow 1$
while $i \leq len(n)$ **do**
  if $i + k >= n$ **then**
    $break$
  end if
  $minval \leftarrow 10$
  $minpos \leftarrow -1$
  for $j : i + 1 \rightarrow i + k + 1$ **do**
    if $n[j] < n[i]$ and $n[j] < minval$ **then**
      $minval \leftarrow n[j]$
      $minpos \leftarrow j$
    end if
  end for
  if $minpos = -1$ **then**
    $ans \leftarrow ans + n[i]$
    $i \leftarrow i + 1$
  else
    $i \leftarrow minpos$
    $k \leftarrow k - (minpos - i)$
  end if
end while
return ans

5 马的遍历问题 (20 分)

给定一个 $n \times m$ 的中国象棋棋盘，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，**写出该算法伪代码**并分析时间复杂度。

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 Algorithm5。

Algorithm 5 $Delete(d)$

Input:

棋盘大小 n, m，起始位置 x, y。

Output:

最终答案 ans。

定义队列 Q
定义答案矩阵 $ans[n][m] = -1$
$pos[8][2] \leftarrow \{(2,1),(1,2),(-1,2),(-2,1),(-2,-1),(-1,-2),(1,-2),(2,-1)\}$
$q.push((x,y))$
while !$Q.empty()$ **do**
    $x \leftarrow Q.front().first$
    $y \leftarrow Q.front().second$
    $Q.pop()$
    for $i : 0 \rightarrow 7$ **do**
        $xx \leftarrow x + pos[i][0]$
        $yy \leftarrow y + pos[i][1]$
        if $xx < 1$ or $xx > n$ or $yy < 1$ or $yy > m$ **then**
            $continue$
        end if
        if $ans[xx][yy] = -1$ **then**
            $ans[xx][yy] \leftarrow ans[x][y] + 1$
            $Q.push(xx, yy)$
        end if
    end for
end while
return ans