计算机学院《算法设计与分析》
(2019 年秋季学期)

第三次作业参考答案

1 最长空位问题 (20 分)

给定一长度为 n 的 01 串 $S =< s_1, s_2, \cdots, s_n >$，你仅有一次机会挑选其中两个元素 $s_i, s_j (1 \leq i, j \leq n)$ 并交换他们的位置。请你设计算法求出交换之后 S 中最多有几个连续的 0，并分析该算法的时间复杂度。

例如，串 S =“10010101” 通过交换 s_4 和 s_7 可以变为 “10000111”，连续的 0 的数量为 4。

解：

记串 S 中 0 的个数为 sum，可知连续的 0 最多不会超过 sum 个。 ……（2 分）

考虑到可以进行一次交换操作，原问题可转化为找出串 S 中最多包含一个 1 的最长子串。该子串的长度即为答案。 ……（5 分）

若在串 S 的首尾分别添加一个字符 1，我们要找的即为在两个 1 之间且仅包含一个 1 的最长子串。枚举所有 1 的位置作为子串中包含的 1 所在的位置，计算第 $i-1$ 个 1 和第 $i+1$ 个 1 之间的长度，从中找出最大值即可。 ……（7 分）

需要注意的是边界情况：若串 S 为 ‘‘1001001′′，我们发现在两个 1 之间且仅包含一个 1 的最长子串是 ‘‘00100′′，但是这里包含的 1 并不能交换到这个子串之外。因此，原问题的答案应为上述最大值与 sum 的较小者。 ……（3 分）

算法的伪代码如 Algorithm 1 所示。

Algorithm 1 $MinChange(S[1..n])$

Input: 长度为 n 的 01 串 S
Output: 交换之后 S 中连续 0 的个数的最大值

```
pos ← < 0 >
for i ← 1 to n do
    if s_i = 1 then
        pos.append(i)
    end if
end for
pos.append(n + 1)
sum ← n − pos.length + 2
ans ← 0
for i ← 1 to pos.length do
    ans ← max{ans, pos_{i+1} − pos_{i−1} − 1}
end for
return min{sum, ans}
```

时间复杂度分析：该算法需枚举 n 个位置，时间复杂度为 $O(n)$。 ……（3 分）

2 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并分析该算法的时间复杂度。

解：

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。 ……（5 分）

因此，在第 i 天升级（购买新机器），这台新机器的总产出为 $(n-i+1) \times X_2$ 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。 ……（12 分）

算法的伪代码请参考 Algorithm 2。

Algorithm 2 $profit(X_1, X_2, U, C, n)$

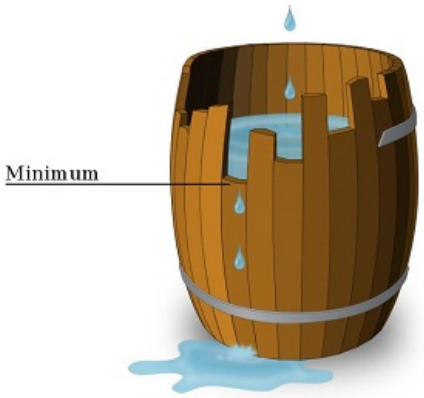
Input: 机器初始产出 X_1，升级的收益 X_2，升级所需成本 U，初始资金 C 以及总天数 n
Output: 该公司 n 天后总资金的最大值

```
for i ← 1 to n do
    if (n − i + 1) × X_2 > U and C ≥ U then
        C ← C − U
        X_1 ← X_1 + X_2
    end if
    C ← C + X_1
end for
return C
```

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 $O(n)$。 ……（3 分）

3 水桶问题 (20 分)

给定 $m = n \times k$ 块木板，第 i 块木板的长度为 a_i，现需要用它们围成 n 个水桶，每个水桶使用 k 块木板。第 j 个水桶的体积 V_j 等于其中最短的那块木板的长度。



为了保持水桶的体积均衡，任两个水桶的体积之差不能超过 l。即是说，这 n 个水桶应满足：

$$|V_x - V_y| \leq l \qquad \forall 1 \leq x, y \leq n$$

请你设计算法判断能否围成这样的 n 个水桶并分析该算法的时间复杂度。

解：

考虑体积均衡的限制，体积最小的水桶一定包含了长度最短的那块木板，其体积与最短的木板的长度相等。为了使其他水桶的体积与最小水桶的体积之差不超过 l，需保证其他水桶中最短的那块木板的长度与所有木板长度的最小值之差不超过 l。因此，仅需判断这些木板中第 n 短的木板长度与最短的木板的长度之差是否小于等于 l。若是，则可将长度最短的 n 块木板分别作

为这 n 个桶的最短木板，剩余的 $(k-1)\times n$ 块木板任意分给这 n 个桶。若不是，则无法围成。……（10 分）

判断的方法可以采用排序 [时间复杂度为 $O(m\log m)$] 或随机化线性时间选择算法 [期望时间复杂度为 $O(m)$]，两种方法均可。……（10 分，其中时间复杂度占 3 分）

这里给出一种使用归并排序来判断的算法伪代码，如 Algorithm 3 所示。

Algorithm 3 $cannikin(a[1..m], n, l)$

Input: 木板的长度 $a[1..m]$，水桶的个数 n，体积限制 l
Output: 是否能围成满足要求的 n 个水桶

```
MergeSort(a[1..m])
if a[n] − a[1] ≤ l then
    return true
else
    return false
end if
```

4 无向图定向问题 (20 分)

给定一个连通无向图 $G=(V,E)$，满足 $|E|=|V|$。

1. 请证明总是存在一种方法对该无向图的每条边进行定向，使得每个点的出度均为 1。(5 分)
2. 请设计一种算法来完成该定向过程并分析该算法的时间复杂度。(15 分)

解：

1. 令 T 为图 G 的任意一棵生成树，易知 T 中所含边数为 $|V|-1$。由于 $|V|=|E|$，故只有一条边没有在该树中，如果将这条边纳入 T 中会形成一个环 C。

   对位于环 C 中的所有边，可以很容易的对它们进行定向，使得每个顶点的出度都为 1。

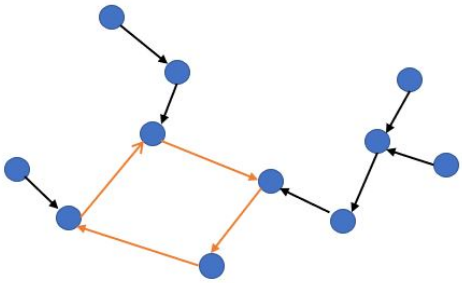


图 1: 满足题目要求的图总可以按照此方法定向（图中橘黄色的边为环 C）

   之后将环中的所有边删去，原图会被分为多棵有根树。对每棵有根树而言，将其中位于环 C 上的点视为根。并将其所有边的方向指定为从儿子节点指向其父亲节点。由于每个点的父亲节点只有一个，因此可以保证每个点的出度都为 1。(不必担心根节点，它们已经在上一步处理过了)

2. 首先在该图上执行 DFS 算法，从而得到一棵 DFS 树。由于 $|V|=|E|$ 并且题目保证图是连通的，故一定只剩下一条边没有在树中。又由于图中的任意一条边要么是树边，要么是反向边，因此剩余的这条边一定是反向边。……（2 分）

   接下来再次遍历 DFS 树 (事实上这一步可以在 DFS 的同时完成)，对每条边 (x,y)（x 是 y 的父亲），将其方向标记为从 y 到 x。

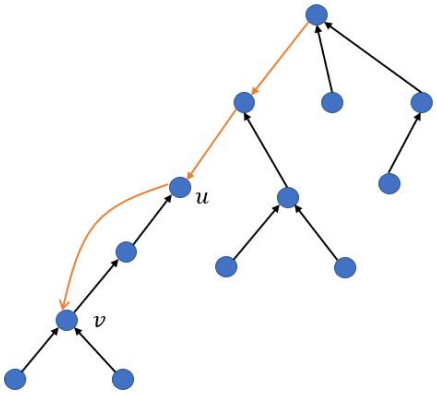


图 2: 通过 DFS 树完成对图的定向，图中橘黄色的树边的方向被反转了

这样一来，除了根节点之外的所有点都已经满足出度为 1 的条件了，同时也还有一条反向边没有被定向。记这条反向边为 (v,u)（u 是 v 的祖先）(如图 2 所示)。将这条边的方向定为从 u 到 v，之后从节点 u 出发，遍历到根节点，将遇到的所有边反向。这样，所有的点包括根节点都满足出度为 1 了。 ……（10 分）

该算法的时间复杂度为 $O(|V|+|E|)=O(|V|)$。 ……（3 分）

5 最短路径问题 (20 分)

在二维平面上有 n 个点，第 i 个点的坐标为 $(x_i,y_i)(1\le i\le n)$。从第 i 个点到第 j 个点的距离为这两点的曼哈顿距离：$d(i,j)=|x_i-x_j|+|y_i-y_j|$。此外，这 n 个点中某些点为传送点，用 $t[i]$ 表示。$t[i]$ 为 0 表示该点不是传送点，$t[i]$ 为 1 表示该点为传送点。任意两个传送点之间的距离均为 0。请设计算法求出在此情况下从点 x 到点 y 的最短路径长度并分析该算法的时间复杂度。

解：

从 i 到 j 有两种可能的方式：

一种为直接从 i 到 j，距离为这两点间的曼哈顿距离。 ……（5 分）

另外一种为从 i 出发，通过某些传送点到达 j。这种情况下，仅需分别找出距离 i 和 j 最近的传送点，分别记为 A,B。则 $i\to A\to B\to j$ 为这种情况下的最短路线。 ……（10 分）

两种情况求得的最短路距离取最小值即为最终答案。 ……（2 分）

算法伪代码如 Algorithm 4 所示。

Algorithm 4 $dist(x[1..n],y[1..n],t[1..n],i,j)$

Input: n 个点的坐标 (x_i,y_i)，传送点标记 $t[1..n]$，给定的两点 i，j
Output: 点 i 到点 j 的最短路径长度

```
dist_A ← ∞
dist_B ← ∞
for k ← 1 to n do
    if t[k] = 1 then
        dist_A ← min{dist_A, |x_i − x_k| + |y_i − y_k|}
        dist_B ← min{dist_B, |x_j − x_k| + |y_j − y_k|}
    end if
end for
ans ← min{|x_i − x_j| + |y_i − y_j|, dist_A + dist_B}
return ans
```

时间复杂度分析：对 i 和 j 分别需要遍历所有点找到距离他们最近的传送点，时间复杂度为 $O(n)$。 ……（3 分）