计算机学院《算法设计与分析》
(2021 年秋季学期)

第四次作业参考答案

1 对下面的每个描述，请判断其是正确或错误，或无法判断正误。对于你判为错误的描述，请说明它为什么是错的。(每小题 5 分，共 20 分)

1. 存在一个 NP 完全问题可以在多项式时间内求解。
2. 如果存在某一 NP 问题不是多项式时间内可求解的，则所有 NP 完全问题都不是多项式时间可求解的。
3. 判定无向图中是否存在环这一问题属于 P 问题。
4. 如果一个问题是 NP-hard，则一定存在一个算法可以在多项式时间内验证该问题的解。

解：

1. 无法判定。
2. 正确。
3. 正确。
4. 错误，NP-hard 问题可能无法在多项式时间内被验证。

2 文件传送问题 (20 分)

给定一个包含 n 个节点的图，图中包含 m 条单向边，起点和终点分别用 s_i 和 t_i 表示。现在需要选取一些节点为发送源，发送源会向外发送数据包，已知数据包只能沿图中单向边进行单向传播，且收到数据包的节点会将数据包通过以该节点为起点的单向边进行转发。

请设计一个高效算法计算最少需要几个发送源才能使所有节点收到数据包，写出该算法伪代码并分析时间复杂度。

1. 问题分析

首先考虑简单情况，即使用 n 个节点，m 条边构成的有向图 $G(V,E)$，假设 G 为有向无环图（DAG），入度为 0 的点无法从其他节点获得数据包，因此答案很明显为入度为 0 的节点数量。

2. 问题求解

当输入图 G 不是有向无环图（DAG）时，对图 G 计算强连通分量，使用强连通分量构造新的有向无环图 G'，将强连通分量作为 G' 的节点，E 中跨越两个强连通分量之间的边作为 G' 的边。

新图 G' 为有向无环图，对于 G' 中入度为 0 的节点，需要指定该强连通分量中任意一个节点为发射源，即可满足题目要求，因此答案为 G' 中入度为 0 的节点个数。

伪代码见4。

3. 时间复杂度分析

计算强连通分量的复杂度为 $O(n+m)$，计算答案仅需遍历所有节点，因此总复杂度为 $O(n+m)$。

Algorithm 1 $scc(G)$

Input: 图 $\boldsymbol{G}$
Output: 强连通分量

$R \leftarrow \{\}$
$G^R \leftarrow G.reverse()$
$L \leftarrow$ DFS(G^R)
$color[1..V] \leftarrow WHITE$
for $i \leftarrow L.length()$ $downto$ 1 **do**
  $u \leftarrow L[i]$
  if $color[u] = WHITE$ **then**
    $L_{scc} \leftarrow$ DFS-Visit(G, u)
    $R \leftarrow R \cup set(L_{scc})$
  end if
end for
return R

Algorithm 2 $dfs(G)$

Input: 图 $\boldsymbol{G}$
Output: 数组 L

新建数组 $\boldsymbol{color[1..V]}$, $\boldsymbol{L[1..V]}$
for $v \in V$ **do**
  $color[v] \leftarrow WHITE$
end for
for $v \in V$ **do**
  if $color[v] = WHITE$ **then**
    $L' \leftarrow$ DFS-Visit(G, v)
    向 $\boldsymbol{L}$ 结尾追加 $\boldsymbol{L'}$
  end if
end for
return L

Algorithm 3 $dfs - visit(G)$

Input: 图 $\boldsymbol{G}$, 顶点 $\boldsymbol{v}$
Output: 按完成时刻从早到晚排列的顶点 $\boldsymbol{L}$

$color[v] \leftarrow GRAY$
初始化空队列 L
for $w \in G.Adj[v]$ **do**
  if $color[w] = WHITE$ **then**
    向 L 追加 DFS-Visit(G, w)
  end if
end for
$color[v] \leftarrow BLACK$
向 $\boldsymbol{L}$ 结尾追加顶点 $\boldsymbol{v}$
return L

Algorithm 4 $file(n, m, (s_i, t_i))$

使用题目中的节点和边构造图 $G(V, E)$
$\{s_1, s_2, \cdots, s_k\} \leftarrow scc(G)$
$V' \leftarrow \{s_1, s_2, \cdots, s_k\}$
$E' \leftarrow \{< s_a, s_b > | < u, v > \in E, u \in s_a, v \in s_b\}$
$ans \leftarrow 0$
for $u : 1 \rightarrow k$ **do**
  $in[s_u] \leftarrow |\{< s_i, s_u > | < s_i, s_u > \in E'\}|$
  if $in[s_u] = 0$ **then**
    $ans \leftarrow ans + 1$
  end if
end for
return ans

3 最小环问题 (20 分)

给定一个包含 n 个点的无向图，边权使用矩阵 $w_{i,j}(w_{i,j} > 0)$ 表示。

请设计一个高效的算法，计算图中最小环（最少包含三个节点）的最小边权和，写出该算法伪代码并分析时间复杂度。

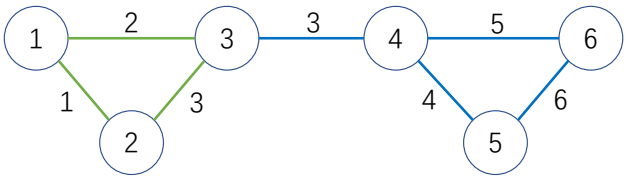

图 1: 无向图最小环

如图1所示，节点 1-2-3 组成的环权重为 6，是该图中的最小环，节点 4-5-6 组成的环权重为 15，不能被称为该图中的最小环，节点 3-4 不满足最少包含三个节点的条件，不能被称为该图中的最小环。

1. 问题分析

为了计算最小环的权重，可以将一个环拆开，例如环 $a - b - \cdots - c - a$，可以拆分为 $a - b$，$a - c$，$b - \cdots - c$，这样仅需枚举节点 a，以及与节点 a 相连的节点 b 和 c，该节点的权重为 $w_{a,b} + w_{a,c} + d_{b,c}$。

由于题目需要满足环最少包含 3 个节点，需要保证路径 $b - \cdots - c$ 不经过节点 a。

回忆上课讲的 Floyd-Warshall 算法，三层循环的最外层 k 表示中间经过的节点最大编号为 k。如果将环中最大的节点 a 拆分，可以通过 Floyd-Warshall 算法的遍历顺序，保证路径 $b - \cdots - c$ 不经过 a。

2. 问题求解

在 Floyd-Warshall 算法求解中，最外层循环的 k 为需要拆分的节点，枚举内层循环 i 和 j 时，判断 k 是否与 i，j 直接相连，如果直接相连，则更新答案 $ans = min(ans, w_{i,k} + w_{j,k} + d_{i,j})$。

伪代码见5。

3. 时间复杂度分析

Floyd-Warshall 算法的时间复杂度为 $O(n^3)$，计算答案的复杂度为 $O(n^3)$，因此总复杂度为 $O(n^3)$。

Algorithm 5 $cycle(n, w[i][j])$

```
d[i][j] ← w[i][j]
ans ← ∞
for k : 1 → n do
    for i : 1 → n do
        for j : 1 → n do
            if i = j or j = k or i = k then
                continue
            end if
            ans ← min(ans, w[i][k] + w[j][k] + d[i][j])
        end for
    end for
    for i : 1 → n do
        for j : 1 → n do
            d[i][j] = min(d[i][j], d[i][k] + d[k][j])
        end for
    end for
end for
return ans
```

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。

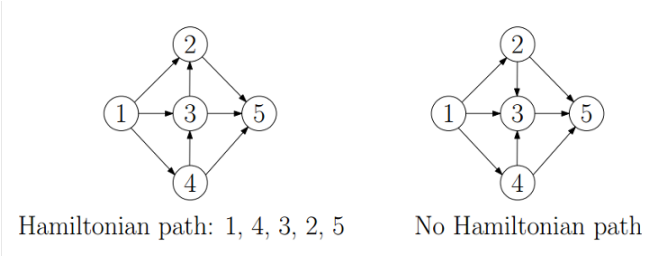

图 2: 哈密顿路径

如图2所示，左侧图包含一条哈密顿路径 1-4-3-2-5，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，写出该算法伪代码并分析时间复杂度。

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 $d[u]$ 表示以点 u 结束的最长路径。

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

初始状态：$d[u] = 0$

为了保证 $d[u]$ 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 $d[u]$ 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7。

3. 时间复杂度分析时间复杂度为 $O(|V| + |E|)$。

Algorithm 6 topo_sort

Input: 图 $\boldsymbol{G}$
Output: 顶点拓扑序

初始化空队列 $\boldsymbol{Q}$
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 7 $hamilton(n, G(V, E))$

$topo \leftarrow topo\_sort(G)$
$ans \leftarrow 0$
$pos \leftarrow -1$
$dp[i] \leftarrow 0$
$b[i] \leftarrow -1$
for $i : 1 \rightarrow n$ **do**
  $pre[a_i] \leftarrow \{u| < u, a_i > \in E, u \in V\}$
  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
$path \leftarrow []$
while $pos \neq -1$ **do**
  $path.append(pos)$
  $pos \leftarrow b[pos]$
end while
return $path$

5 二分图判定问题 (20 分)

二分图是指一个无向图 $G = (V, E)$, 它的所有顶点可被分成两个子集，且同一个子集中任何两顶点间都没有边相连。（换言之，G 为二分图，当且仅当存在两个集合 V_1, V_2 满足 $V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset$，$E$ 中每条边都连接了 V_1 中某个点与 V_2 中某个点。）

1. 请证明：二分图中不存在长度为奇数的环。（5 分）
2. 请设计一个基于广度优先搜索 (BFS) 的算法来判断无向图 G 是否为二分图，写出该算法伪代码并分析该算法时间复杂度。（15 分）

解：

1. 证明

令 $G = (V, E)$ 为一个二分图，V_1, V_2 为题目所述的两个集合。考虑图 G 中的任意一个环 (记为 $C =< v_1, v_2, v_3, \ldots, v_n, v_1 >$)，不失一般性，我们假设 $v_1 \in V_1$，根据二分图的性质，可以得出结论：对于该环中任意一点 v_i，如果 i 是奇数，则 $v_i \in V_1$，如果 i 是偶数，则 $v_i \in V_2$。又因为存在一条边 (v_n, v_1)，故 n 一定是偶数。因此，G 中任意一个环的长度一定为偶数，不存在长度为奇数的环。

2. 二分图判定

1) 问题求解二分图判定的最简单的方法是染色法，即利用二分图性质可以将所有节点分为两个集合 V_1 和 V_2，满足 $V_1 \cap V_2 = \emptyset$，$V_1 \cup V_2 = V$，并且点集中不存在边。可以将 V_1 中的点染为红色，将 V_2 中的点染为绿色，每条边两侧的节点颜色不通，利用广度优先搜索（BFS）将图染色，若能成功染色，则该图为二分图，否则不是二分图。

伪代码见算法8。

2) 时间复杂度分析

该算法遍历了所有节点，时间复杂度为 $O(|V| + |E|)$。

Algorithm 8 $bipartite(n, G(V, E))$

```
color[n] ← −1
for i : 1 → n do
    if color[i] >= 0 then
        continue
    end if
    初始化队列 q
    color[i] ← 0
    q.push(i, 0)
    while 队列 q 不为空 do
        u, c ← q.front()
        q.pop()
        adj ← {v| < u, v >∈ E}
        for i : 1 → |adj| do
            if color[adj[i]] = −1 then
                color[adj[i]] = 1 − c
                q.push(adj[i], 1 − c)
            end if
            if color[adj[i]] = c then
                return false
            end if
        end for
    end while
end for
return true
```