计算机学院《算法设计与分析》
(2022 年秋季学期)

第四次作业参考答案

作业提交截止时间：2022 年 12 月 19 日 23 : 55

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/无法判断的描述，请说明它为什么是错误/无法判断的。(每小题 5 分，共 20 分)

1. P 类问题为 NP 类问题的真子集。
2. 如果假设 $P \neq NP$，则 NP 完全问题可以在多项式时间内求解。
3. 若 SAT 问题可以用复杂度为 $O(n^9)$ 的算法来解决，则所有的 NP 完全问题都可以在多项式时间内被解决；
4. 对于一个 NP 完全问题，其所有种类的输入均需要用指数级的时间求解。

解：

1. 无法判定,P 是否等于 NP 是开放问题。
2. 错误，如果 NP 完全问题可以在多项式时间内求解，则所有 NP 问题可以在多项式时间内求解，与假设 $P \neq NP$ 矛盾。
3. 正确；
4. 错误。虽然 SAT 是 NP 完全问题，但是对于所有满足 $2-SAT$ 条件的输入，都可以在多项式时间内求解。

2 颜色交错最短路问题 (20 分)

给定一个无权有向图 $G=<V,E>$（所有边长度为 1），其中 $V=\{v_0,v_1,...v_{n-1}\}$，且这个图中的每条边不是红色就是蓝色（$\forall e \in E, e.color = red$ 或 $e.color = blue$），图 G 中可能存在自环或平行边。

现给定图中两点 v_x, v_y，请设计算法求出一条从 v_x 到 v_y，且红色和蓝色边交替出现的最短路径。如果不存在这样的路径，则输出-1。请给出**分析过程、伪代码以及算法复杂度**。

解：

1. 问题分析

注意到，合法路径应当满足红色和蓝色边交替出现。而并不知道最短路径的第一条边的颜色，故两种情形都要加以考虑。

2. 广度优先搜索

故我们考虑每个点实际上有两种状态 0/1，分别表示到达该点时使用了蓝色边的最短路径长度，和到达该点时使用了红色边的最短路径长度。然后利用广度优先搜索，每个点都会被其最早一次可扩展的机会扩展即可。

3. 时间复杂度分析

注意到总的状态个数为 $O(|V|)$ 级别的，而每次搜索新的状态都是在枚举点所链接的边，故总的时间复杂度为 $T(|V|,|E|)=O(|V|+|E|)$。

参考伪代码如1所示。

Algorithm 1 $shortest-path(V,E,v_x,v_y)$

Input: 给定的无权有向图 $G(V,E)$

Output: 从 v_x 到 v_y 的颜色交错最短路径。

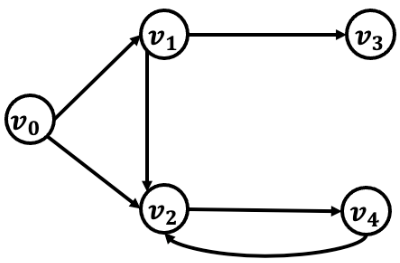
$vis[v_i][0/1] \leftarrow 0$
$dis[v_x][0/1] \leftarrow 0$
$vis[v_x][0/1] \leftarrow 1$
$to[v_i][0]=\{v_j| <v_i,v_j>\in E, <v_i,v_j>.color=red\}$
$to[v_i][1]=\{v_j| <v_i,v_j>\in E, <v_i,v_j>.color=blue\}$
将 $(v_x,0),(v_x,0)$ 加入搜索队列 Q
while Q 不为空 **do**
  将 Q 队首状态取出至 (now,col)
  for $nxt:to[now][1-col]$ **do**
    if $vis[nxt][1-col]=0$ **then**
      $vis[nxt][1-col]=1$
      $dis[nxt][1-col]=dis[now][col]+1$
      将 $(nxt,1-col)$ 加入搜索队列 Q
    end if
  end for
end while
if $vis[v_y][0]=0 \vee vis[v_y][1]=0$ **then**
  return -1
else if $vis[v_y][0]=0$ **then**
  return $dis[v_y][1]$
else if $vis[v_y][1]=0$ **then**
  return $dis[v_y][0]$
else
  return $\min(dis[v_y][0],dis[v_y][1])$
end if

3 最小闭合子图问题 (20 分)

对于一个有向图 $G=<V,E>$，其**闭合子图**是指一个顶点集为 $V'\subseteq V$ 的子图，且保证点集 V' 中的所有出边都还指向该点集。换言之，V' 需满足对所有边 $(u,v)\in E$，如果点 u 在集合 V' 中，则点 v 也一定在集合 V' 中。

现给定一个包含 n 个点的有向图 $G=<V,E>$，请设计算法求出该图中的闭合子图至少应包含几个顶点，并分析其时间复杂度。

例如,给定如下图所示的包含 5 个顶点的图,其闭合子图可能为:$\{v_3\},\{v_0,v_1,v_2,v_3,v_4\},\{v_2,v_4\}$。最小的闭合子图仅包含 1 个顶点，为 $\{v_3\}$。请给出**分析过程、伪代码以及算法复杂度**。



解：

1. 问题分析

对于任意一点，若其在闭合子图中，则其所有出边节点都在闭合子图中，故递推之，其所在的强连通分量也必在该闭合子图中。

2. 闭合子图实质

故我们可以考虑将每个强连通分量视作单一的点，不同强连通分量包含的节点有连边则这两个强连通分量连边。对于这样收缩出来的图 G' 是一个有向无环图。而在这之上选择闭合子图，即选择任意节点后，要将其所有后继节点都选入闭合子图。

3. 贪心策略

故我们在 G' 中可以直接选取所有没有后继节点的所有强联通分量中，包含原图节点最少的那一个作为闭合子图，即为所求。

4. 时间复杂度分析

注意到求解强连通分量的过程是 $O(|V|+|E|)$ 的。而贪心选取即遍历所有强连通分量，故总时间复杂度为 $T(|V|,|E|)=O(|V|+|E|)$。

参考伪代码如5所示。

Algorithm 2 $scc(G)$

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 3 $dfs(G)$

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

新建数组 $\boldsymbol{color[1..V], L[1..V]}$
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 4 $dfs - visit(G)$

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 5 $mingraph(V, E)$

Input: 给定的图 $G(V, E)$

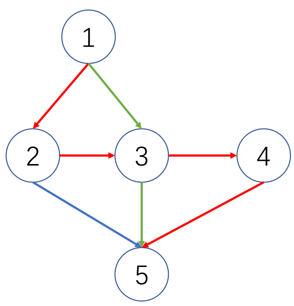
Output: 所选出的最小闭合子图

$\{s_1, s_2, \cdots, s_k\} \leftarrow scc(G)$
$V' = \{s_1, s_2, \cdots, s_k\}$
$E' = \{< s_u, s_v > \mid < u, v > \in E, u \in s_u, v \in s_v\}$
$out[s_i] = |\{< s_i, s_u > \mid < s_i, s_u > \in E'\}|$
$ANS \leftarrow \emptyset$
for $i : 1 \rightarrow k$ **do**
  if $out[s_i] == 0$ **then**
    if ANS 为空或者 $|s_i| \leq |ANS|$ **then**
      $ANS \leftarrow s_i$
    end if
  end if
end for
return ANS

4 食物链问题 (20 分)

给定一个食物网，包含 n 个动物，m 个捕食关系，第 i 个捕食关系使用 (s_i, t_i) 表示，s_t 捕食者，t_i 表示被捕食者，根据生物学定义，食物网中不会存在环。

长度为 k 的食物链指包含 k 个动物的链：$a_1, a_2, \cdots, a_k$，其中 a_i 会捕食 a_{i+1}，一个食物链为**最大食物链**当且仅当 a_1 不会被任何动物捕食，且 a_k 不会捕食任何动物。



如图上图所示，该食物网存在 5 个动物，7 个捕食关系，其中红色和绿色均可以称为最大食物链，而蓝色食物链则不能成为最大食物链，图中一共包含 5 个最大食物链，分别是 1-2-5，1-2-3-5，1-2-3-4-5，1-3-5，1-3-4-5。

请设计一个高效算法计算食物网中最大食物链的数量，并给出**分析过程、伪代码以及算法复杂度**。

1. 问题分析

使用 n 个动物作为节点 V，m 个捕食关系作为边 E，构造图 $G(V,E)$。

最大食物链可以表示为图上的一条路径，该路径满足起点的入度为 0，终点的出度为 0，因此可以考虑在图 G 上进行动态规划以求解问题。

2. 问题求解

状态定义：使用 $dp[i]$ 表示编号为 i 的节点为路径终点的数量。

转移方程：对于前驱节点 $pre[v]=\{u|<u,v>\in E, u\in V\}$，$dp[v]=\sum_{u\in pre[v]} dp[u]$。

边界条件：对于所有入度为 0 的点 u，$dp[u]=1$。

遍历顺序：以图 G 拓扑排序顺序进行遍历。

伪代码见算法7。

3. 时间复杂度分析

对于每个节点的转移复杂度为 $|pre[v]|$，总复杂度为 $O(n+m)$。

Algorithm 6 topo_sort

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 7 $foodchain(n, m, (s_i, t_i))$

使用推荐关系 (s_i, t_i) 构造图 $G(V,E)$
$topo \leftarrow topo\_sort(G)$
$ans \leftarrow 0$
for $i : 1 \rightarrow n$ **do**
  $pre[a_i] \leftarrow \{u|<u,a_i>\in E, u\in V\}$
  $out[a_i] \leftarrow |\{u|<a_i,u>\in E, u\in V\}|$
  $dp[a_i] \leftarrow \sum_{u\in pre[a_i]} dp[u]$
  if $out[a_i] = 0$ **then**
    $ans \leftarrow ans + dp[a_i]$
  end if
end for
return ans

---

5 景区限流问题问题 (20 分)

已知某市有热门景区 m 个，可以表示为集合 $S=\{s_1,s_2,...,s_m\}$，有游客 n 人，可以表示为 $T=\{t_1,t_2...,t_n\}$。每名游客有 k 个心仪的景区，但每个景区最多容纳 l 人，问最多有多少人能够去到自己心仪的景区，并给出**分析过程、伪代码以及算法复杂度**。

对于游客与景区的偏好关系，用 H 表示，则 H 可以表示如下形式，其中 (t_u,s_v) 表示游客 t_u 偏好景点 s_v，。

$$H=\begin{Bmatrix} (t_1,s_{11}), & \cdots & ,(t_1,s_{1k}), \\ \vdots & \ddots & \vdots \\ ,(t_n,s_{n1}), & \cdots & ,(t_n,s_{nk}) \end{Bmatrix}$$

1. 问题分析

该问题可以建模为二分图的最大匹配问题，其中游客为左端点，而景区为右端点。

2. 问题求解

由于每个景区可以容纳多名游客，因此可以将右端点 R 扩展到 $m*l$ 个，其中景区 s_i 对应点 $r_i,...,r_{(m-1)l+i}$。相应的，游客 t_j 对应的喜好扩展到 $(t_u,r_i),...,(t_u,r_{(m-1)l+i})$。

那么可以构建二分图 $G=(L,R,E)$，其中 $L=T$,R 为 S 扩展后集合，E 为扩展后的边集，对该二分图进行最大匹配得到的条数即为结果。

伪代码见10

3. 时间复杂度分析

匈牙利算法的时间复杂度是 $O(|V|\times|E|)$, 对扩展后的二分图即 $O((n+ml)\times nk)$，而扩展的时间复杂度为 $O((m+n)k)$，因此时间复杂度是 $O((n+ml)\times nk)$，若将 k,l 视为常数，则为 $O(n^2+mn)$

Algorithm 8 $hungarian(G)$

Input: 二分图 $\boldsymbol{G}=<\boldsymbol{L},\boldsymbol{R},\boldsymbol{E}>$
Output: 匹配数组 $\boldsymbol{matched}$

```
新建一维数组 matched[1..|R|], color[1..|R|]
for u ∈ R do
    matched[u] ← NULL
end for
for v ∈ L do
    //初始化 color 数组
    for u ∈ R do
        color[u] ← WHITE
    end for
    DFS-Find(G, v)
end for
return matched
```

Algorithm 9 $DFS - Find(G)$

Input: 二分图 $\boldsymbol{G =< L, R, E >}$, 顶点 $\boldsymbol{v}$

Output: 是否存在从顶点 $\boldsymbol{v}$ 出发的交替路径

//深度优先搜索寻找以顶点 $\boldsymbol{v}$ 出发的交替路径
for $u \in G.Adj[v]$ **do**
    if $color[u] = BLACK$ **then**
        continue
    end if
    $color[u] \leftarrow BLACK$
    if $matched[u] = NULL$ or DFS-Find($G, matched[u]$) **then**
        $matched[u] \leftarrow v$
        return $True$
    end if
end for
return $False$

Algorithm 10 scene

Input: 景区集合 S, 游客集合 T, 游客心仪景区关系 H, 个人心仪景点数 k, 景点最大容纳游客数 l

Output: 最大匹配

//根据 S, T, H 生成二分图 $G = (L, R, E)$
$L \leftarrow T$
for $s_i \in S$ **do**
    for $j : 1- > k$ **do**
        将 $r_{(j-1)l+i}$ 加入集合 R
    end for
end for
for $(t_u, s_v) \in E$ **do**
    for $j : 1- > k$ **do**
        将 $(t_u, r_{(j-1)l+v})$ 加入集合 E
    end for
end for
$G \leftarrow (L, R, E)$
$res \leftarrow |hungarian(G)|$
return res