# 高等理工学院《算法设计与分析》
# (2020 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 4 分，共 20 分)

1. $P = NP$;
2. $UNSAT \in NP$ ($UNSAT$ 是指，给定一个布尔表达式 $\phi$，判断是否对其中变量的所有取值，$\phi$ 的值均为 $false$);
3. $NP\text{-}hard \subseteq NP$;
4. 给定一个包含 $n$ 个点的图 $G$，判断其中是否包含大小为 10 的团不能在多项式时间内被解决;
5. 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。
2. 无法判定。
3. 错误，$NP\text{-}hard$ 问题未必多项式时间内可验证;
4. 错误，该问题仅需 $O(\binom{n}{10})$ 次判断即可得出结果。
5. 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 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。

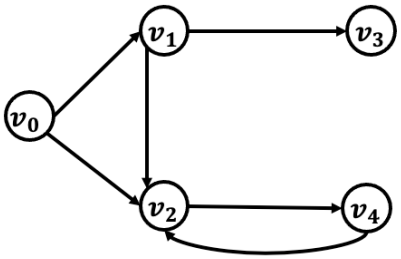
例如,给定如下图所示的包含 5 个顶点的图,其闭合子图可能为:$\{v_3\}, \{v_0, v_1, v_2, v_3, v_4\}, \{v_2, v_4\}$。最小的闭合子图仅包含 1 个顶点，为 $\{v_3\}$。

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1所示。

**Algorithm 1** $mingraph(V,E)$

**Input:** 给定的图 $G(V,E)$
**Output:** 所选出的最小闭合子图

求出 $G$ 图的所有强连通分量 $\{s_1,s_2,\cdots,s_k\}$
$V'=\{s_1,s_2,\cdots,s_k\}$
$E'=\{< s_u,s_v > | < u,v >\in E,u\in s_u,v\in s_v\}$
$out[s_i]=|\{< s_i,s_u > | < s_i,s_u >\in E'\}|$
$ANS\leftarrow\emptyset$
**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** $ANS$

# 3 字母顺序问题 (20 分)

现有一种全新语言，其仍使用小写英文字母，但这门语言的字母顺序未知。现给定一个这门语言的顺序列表，该顺序列表内的单词已经按照这门新语言的字母顺序进行了排序 (以英语的字母顺序为例，那么对于一个有 $bad$ 三个字母的单词，对应在顺序单词列表中的单词会是 $abd$, 因为按照英文的顺序 $a$ 先于 $b$ 先于 $d$)。

请设计算法，根据这个输入的列表，还原出该语言中已知的字母顺序。并分析该算法的时间复杂度。

例如，对于输入列表 $[wrt,wrf,er,ett,rftt]$，我们应输出 $wertf$。

解:

**1. 问题分析**

记输入列表为 $s_i$, 注意到对于任意两个单词 $s_i,s_j(i<j)$，其第一个不同的字母出现在第 $k$ 位，即得到了 $s_i[k]<s_j[k]$ 的大小关系。则可以建立有向边 $< s_i[k],s_j[k] >$ 表示字母的先后顺序。

**2. 拓扑排序**

故在此有向图上做一次拓扑排序，就可以得出已知字母的顺序。

**3. 时间复杂度分析**

记单词总数为 $n$，出现的字母总数为 $m$，第 $i$ 个单词的长度为 $s_i$。

考虑到建立所需要的边需要对每个单词遍历和其他单词的第一个不同字母，而建立出来的图是以字母为节点，以单词顺序关系为边，故总时间复杂度为 $T(n, m, \{s_i\}) = O((n \times \sum_{i=1}^{n} s_i) + n^2 + m)$。

参考伪代码如2所示。

**Algorithm 2** $sort(\{s_i\})$

**Input:** 给定的 $n$ 个排好序的单词 $s_1 \sim s_n$

**Output:** 还原出的已知顺序

```
E ← ∅
V ← ∅
for i : 1 → n do
    for j : i + 1 → n do
        k ← s_i, s_j 第一个不同字母的位置
        < s_i[k], s_j[k] > 加入 E 集合
        s_i[k], s_j[k] 加入 V 集合
    end for
end for
对图 G(V, E) 求解拓扑序 T。
return T
```

# 4 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。

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3所示。

# 5 删边问题 (20 分)

给定一包含 $n$ 个点 $n$ 条边的无向图 $G =< V, E_g >$，其中 $E_g$ 以有顺序的数组形式给出，$E_g = [e_1, \cdots, e_n]$。此图是在一棵树 $T =< V, E_t >$ 的基础上加上一条边 $e_{add}$ 构成的。

现请设计算法试求出边 $e \in E_g$，使得删掉该边后的图 $G' =< V, E_g - \{e\} >$ 仍是一棵树 (如果有多条边满足该条件，请输出在 $E_g$ 的列表中顺序最靠前的一条边)。

解：

**1. 问题分析**

注意到需要删掉的边，就是在该图唯一的环上出现在列表中最早的一条边。

**Algorithm 3** $shortest-path(V,E,v_x,v_y)$

**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**

**2. kruskal 算法**

回忆 kruskal 算法，会删除图上所有环中权值最大的一条边，故我们可以倒序注意考虑 $E_g$ 的每一条边，这样 kruskal 算法就会删去环中最早出现的一条边。

**3. 时间复杂度分析**

注意到，这里的 kruskal 算法省去了排序的复杂度，故总时间复杂度分析为 $T(|E|,|V|) = O(|E|\alpha(|V|))$。

参考伪代码如4所示。

**Algorithm 4** $cut-edge(V,E_g)$

**Input:** 给定的无向图 $G(V,E_g)$
**Output:** 需要删除的边 $e$

**for** $i : n \to 1$ **do**
  $x \leftarrow e_i.x$
  $y \leftarrow e_i.y$
  **if** $x, y$ 不在同一个集合内 **then**
    将 $x, y$ 所在的两个集合合并
  **else**
    **return** $e_i, i$
  **end if**
**end for**