

除非特别指出，本次作业中所有的图默认为简单图。

**Q1.**  $((15 + 10) + (10 + 10) = 45$  分)

本题不考虑图中存在负环的情况。

(1) 某同学在 Bellman-Ford 算法的基础上提出了一种基于队列的优化方案，并将其命名为“SPFA” (Shortest Path Faster Algorithm)，算法1给出了 SPFA 的伪代码：

---

**Algorithm 1:** Shortest Path Faster Algorithm

---

**Input:**  $G = \langle V, E \rangle, w : E \rightarrow \mathbb{R}, s \in V$

---

```

1 INITIALIZE-SINGLE-SOURCE( $G, s$ );
2  $Q \leftarrow \{s\}$ ;
3 while  $Q \neq \emptyset$  do
4    $u \leftarrow \text{DEQUEUE}(Q)$ ;
5   foreach  $\langle u, v \rangle \in G.E$  do
6     if  $v$  can be relaxed by  $u$  then
7       RELAX( $u, v, w$ );
8       if  $v \notin Q$  then
9         ENQUEUE( $Q, v$ );
10      end
11    end
12  end
13 end
```

---

(1.1) 我们将第一次 while 循环过程称为“第一轮”，第一轮中入队节点出队的全过程称为“第二轮”，以此类推。记  $s$  到  $v$  的所有最短路径中，所含节点数最少的路径的节点数为  $\varphi(s, v)$ ，试证明： $s$  到  $v$  的最短路径的长度能够在不超过第  $\varphi(s, v) - 1$  轮中被确定。

显然，此引理不仅直接导出了 SPFA 的正确性，同时也给出了  $O(|V||E|)$  的最坏时间复杂度上界。

(\*1.2) 尝试构造正权图列  $\{G_{n,m}\}, m \in \left[n - 1, \frac{n(n-1)}{2}\right]$ ，满足： $|V_{n,m}| = \Theta(n), |E_{n,m}| = \Theta(m)$ ，使得 SPFA 在  $G_{n,m}$  上的运行时间  $T(n, m) = \Omega(nm)$

提示：参考形如图1的图，构造恰当的边权使得节点  $t$  入队  $\Theta(n)$  次。

这一构造说明：对于正权图而言，SPFA 在最坏情况下的性能严格劣于 Dijkstra 算法。

(2) 某同学在 Dijkstra 算法的基础上提出了一种能够处理有负权的图的变种算法，其伪代码见算法2。

---

**Algorithm 2:** Reentrant Dijkstra Algorithm

---

**Input:**  $G = \langle V, E \rangle, w : E \rightarrow \mathbb{R}, s \in V$

```

1 INITIALIZE-SINGLE-SOURCE( $G, s$ );
2  $S \leftarrow \emptyset$ ;
3  $Q \leftarrow G.V$ ;
4 while  $Q \neq \emptyset$  do
5      $u \leftarrow \text{EXTRACT-MIN}(Q)$ ;
6      $S \leftarrow S \cup \{u\}$ ;
7     foreach  $\langle u, v \rangle \in G.E$  do
8         if  $v$  can be relaxed by  $u$  then
9             RELAX( $u, v, w$ );
10            if  $v \notin S$  then
11                DECREASE-KEY( $Q, v, v.d$ );
12            else
13                 $S \leftarrow S \setminus \{v\}$ ;
14                INSERT( $Q, v$ );
15            end
16        end
17    end
18 end

```

---

(2.1) 对比算法2与原始 Dijkstra 算法，简要说明为什么算法2能够处理有非正权边的图。

(\*2.2) 尝试构造图列  $\{G_n\}$ ，满足： $|V_n| = \Theta(n), |E_n| = \Theta(n)$ ，使得算法2在  $G_n$  上的运行时间  $T(n) = \Omega(2^n)$

提示：考虑形如图2的图，构造恰当的边权使得算法运行过程中出现大量“回溯”。

这一构造说明：对于允许有非正权边的图，算法2在最坏情况下的性能严格劣于 Bellman-Ford 算法。

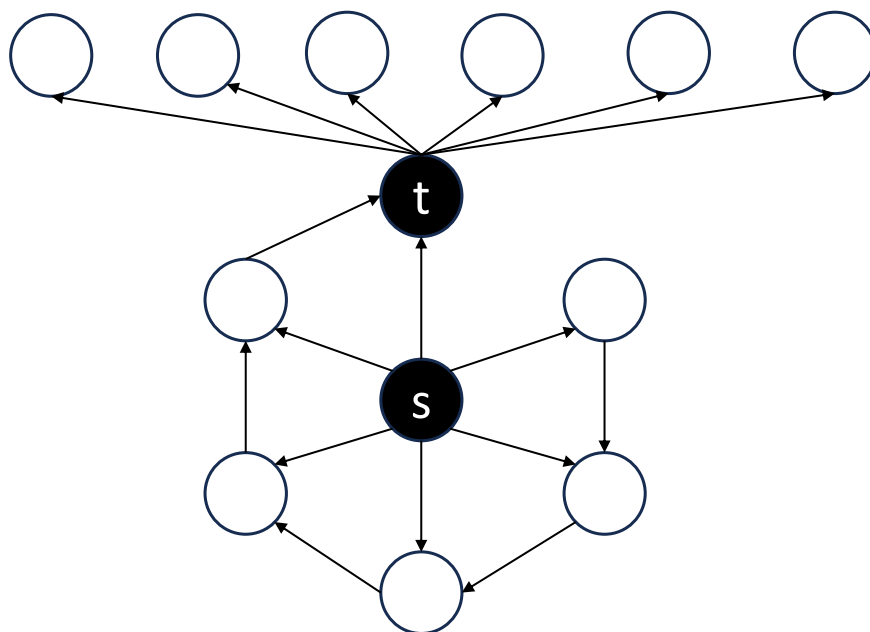


图 1: SPFA 在正权图上的较差情况示例

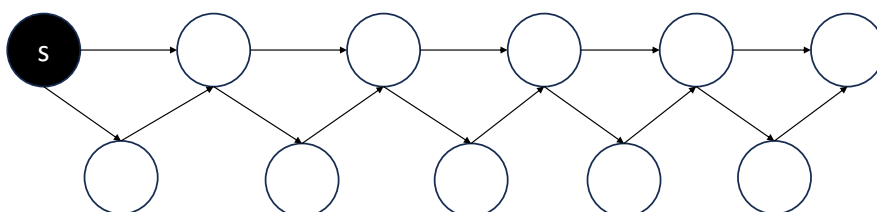


图 2: 算法2的较差情况示例

**Q2.**  $((15 + 10) + 15 + (5 + 5 + 5) = 55$  分)

给定图  $G = \langle V, E \rangle$  以及深度优先森林  $G_\pi = \langle V, E_\pi \rangle$ ，记节点  $v$  在深度优先遍历中的发现时间戳为  $\text{dfn}[v]$ ， $G_\pi$  中以  $v$  为根的子树的节点集为  $S_v$ ， $S_v$  中的节点经一条非树边可达的节点集为  $T_v$ ， $T_v$  中可达  $v$  的节点集为  $P_v$ 。

定义： $\text{low}[v] = \min_{u \in \{v\} \cup P_v} \text{dfn}[u]$

(1) 本问中的深度优先遍历以这样的顺序进行：起始点为节点 1，每当有多个节点可供选择时，总是按照编号**从小到大的顺序**进行选择。

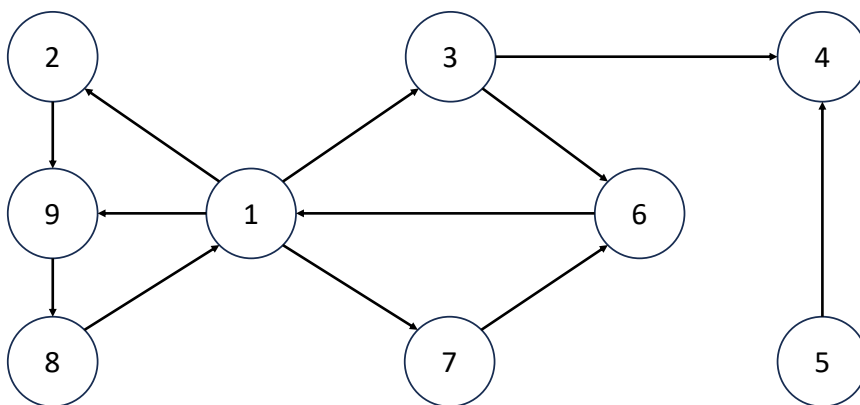


图 3: 示例图

(1.1) 对图3进行深度优先遍历，标注出每条边属于哪种类型（树边、后向边、前向边、横向边），写出各节点的  $\text{dfn}$  值以及  $\text{low}$  值，找出图中的所有强连通分量并以有向无环图的形式进行表示。

(1.2) 将图3中的有向边改为无向边并进行深度优先遍历，标注出每条边属于哪种类型（树边、后向边、前向边、横向边），写出各节点的  $\text{dfn}$  值以及  $\text{low}$  值，找出图中的所有割点和桥。一个点（一条边）是割点（桥）是指：将它删除会导致图的连通分量数量增加。

(\*2) 设计线性最坏时间复杂度的算法计算所有节点的  $\text{low}$  值。

提示：对于有向图，需要使用某种数据结构维护“已被发现的节点中，有哪些可达当前节点”。

(\*3)  $\text{low}$  数组可用于解决许多与连通性有关的问题。

(\*3.1) 简要描述如何利用  $\text{low}$  数组找出有向图中的所有强连通分量。

(\*3.2) 简要描述如何利用  $\text{low}$  数组找出无向图中的所有割点。

(\*3.3) 简要描述如何利用  $\text{low}$  数组找出无向图中的所有桥。

**Q3.** (10 + (15 + 10) + 15 = 50 分)

本题中的“环”定义为“至少包含三条边的简单回路”，环的“大小”定义为边权和。本题仅考虑正权图的情况。

(\*1) 假设我们希望求出有向图的全局最小环。某同学注意到：记环中编号最大的节点为  $w$ ，环中指向  $w$  的节点为  $v$ ， $w$  指向的节点为  $u$ ，则环是由  $\langle v, w \rangle$ 、 $\langle w, u \rangle$ ，以及一条  $u$  到  $v$  的路径拼成的。根据此思路设计算法，并分析最坏时间复杂度。

(2) 假设我们需要求出有向图中经过某个点的最小环。某同学注意到：任何一个经过  $s$  的环都是由一条入边  $\langle t, s \rangle$  以及一条  $s$  到  $t$  的路径拼成的。

(2.1) 若图中不存在反向平行边，根据上述思路设计算法，并分析最坏时间复杂度。

(2.2) 若图中可能存在反向平行边，完善你在 (2.1) 中设计的算法，并分析最坏时间复杂度。

(\*3) 给定无向连通图  $G = \langle V, E \rangle$ ，满足  $|E| - |V| = k$  ( $k \in \mathbb{N}$ ，将  $k$  视为常数)，设计（关于  $|V|$  的）线性最坏时间复杂度算法求出全局最小环。

**Q4.**  $((10 + (10 + 15)) + (10 + 5) = 50$  分)

布尔适定性问题与图论问题具有紧密的联系。

若一个合取范式的每个子句中恰有  $k$  个不同的“文字”，则称其为“ $k$ -CNF”。“文字”是指一个布尔变量或一个布尔变量的否定。 $k$ -SAT 问题是指：给定一个  $k$ -CNF，判断它是否可满足（即是否存在成真赋值）。

例如： $(x \vee y) \wedge (\neg y \vee z) \wedge (\neg z \vee \neg x)$  是一个含有 3 个子句的 2-CNF， $x, y, z$  是变量， $x, \neg x, y, \neg y, z, \neg z$  是文字。当  $x$  为真， $y, z$  为假时，整个公式为真，因此它是可满足的。

(1) 任意一个 2-CNF 都可以被转化为有向图的形式，假设子句数量为  $n$ ，变量数量为  $m$ 。

首先，对于每个子句  $p \vee q$ ，将其改写为  $(\neg p \rightarrow q) \wedge (\neg q \rightarrow p)$ ，这样就得到了一个由  $2n$  个蕴含式相与所构成的布尔公式。

然后，建立一个含有  $2m$  个节点的图，每个文字对应一个节点。对于前一步中每个子句  $p \rightarrow q$ ，在图中添加有向边  $\langle p, q \rangle$ ，当出现重边时，只保留其中一条。记所得到的有向图为  $G$ 。

(1.1) 将上面给出的 2-CNF 转化为有向图。

(\*1.2) 记  $G$  的所有强连通分量构成了有向无环图  $G^{\text{SCC}}$ ，对  $G^{\text{SCC}}$  进行拓扑排序。对于  $G$  中的每个节点  $p$ ，定义：

$$\text{top}[p] = p \text{ 所属的强连通分量在 } G^{\text{SCC}} \text{ 中的拓扑序}$$

(\*1.2.1) 证明：若  $\exists p, \text{top}[p] = \text{top}[\neg p]$ ，则原公式不可满足。

(\*1.2.2) 若  $\forall p, \text{top}[p] \neq \text{top}[\neg p]$ ，设计一个线性最坏时间复杂度的算法，借助  $\text{top}$  数组的值构造原公式的一个成真赋值。你需要证明算法的正确性。

(2) 任意一个 3-CNF 都可以转化为无向图的形式，假设子句数量为  $n$ 。

首先，为每个子句中的每个文字建立一个节点，并且将同一个子句中的文字相连，此时图中一共有  $n$  个“三角形”。

然后，将不同子句中互为否定的文字相连，例如图中如果有 3 个  $x$  和 5 个  $\neg x$ ，且它们所在的子句各不相同，那么它们之间一共要连 15 条边。

(2.1) 将  $(\neg x \vee y \vee z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z)$  转化为无向图。

(2.2) 找出 (2.1) 所得到的图中的 3 个节点，使得它们两两不相邻，并通过这 3 个点给出原公式的一个成真赋值。

以上转化事实上构成了 3-SAT 问题到独立集问题的多项式时间归约。