提高算法班

有向图的连通性

Mas

连通

无向图

对于一张无向图 $G = (V, E)$, 对于 $u, v \in V$ 若存在一条路径使得 $v_0 = u, v_k = v$, 则称 u 和 v 是 **连通的 (Connected)**

若 G 满足其中任意两个顶点均连通,则称 G 是 **连通图 (Connected graph)**

若 H 是 G 的一个连通子图

且 G 不存在 F 满足 $H \subsetneq F \subseteq G$ 且 F 为连通图, 则 H 是 G 的一个 **连通块/连通分量(Connected component)**(极大连通子图)

有向图

对于一张有向图 $G = (V, E)$, 对于 $u, v \in V$, 若存在一条途径使得 $v_0 = u, v_k = v$ 则称 u **可达** v

若一张有向图的节点两两互相可达 , 则称这张图是 **强连通的 (Strongly connected)**

若一张有向图的边替换为无向边后可以得到一张连通图,则称原来这张有向图是 **弱连通的 (Weakly connected)**

与连通分量类似 , 也有 **弱连通分量 (Weakly connected component)**和 **强连通分量 (Strongly Connected component)**

DFS生成树

有向图的 DFS 生成树主要有 4 种边(不一定全部出现):

树边

右图中灰色边，当搜索遇到一个还没有访问过的结点时就形成一条树边

反祖边

右图中红色边 (7 → 1)，指向祖先结点的边

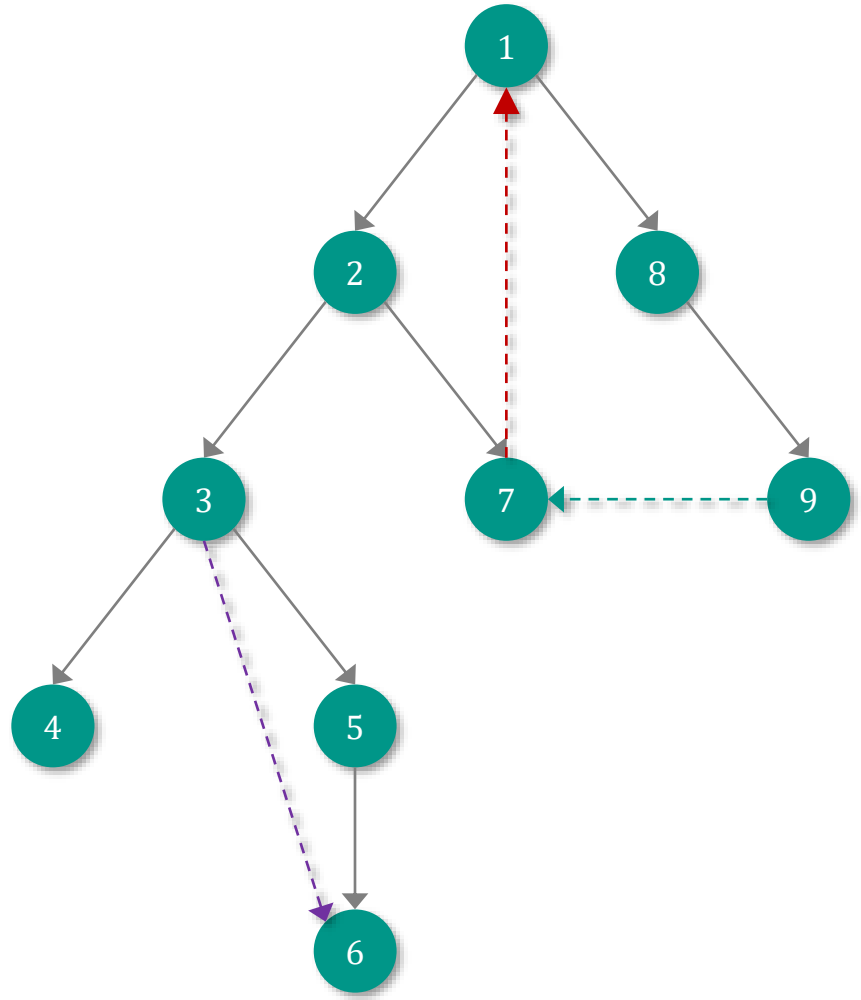
横叉边

右图中绿色边 (9 → 7)

在搜索的时候遇到了一个已访问过的结点，但是该结点 **并不是** 当前结点祖先

前向边

示意图中紫色边 3 → 6，在搜索时候遇到子树中的结点时形成



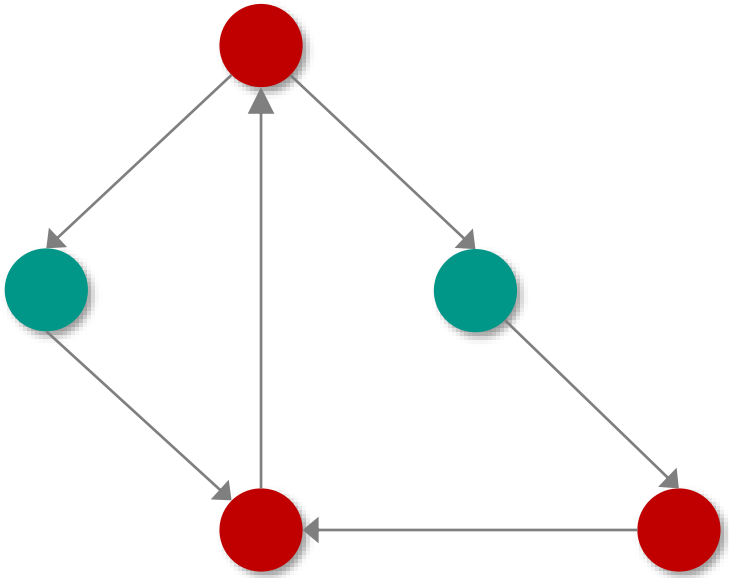
DFS生成树

考虑 DFS 生成树与强连通分量之间的关系

若结点 u 是某个强连通分量在搜索树中遇到的第一个结点

那么这个强连通分量的其余结点肯定是在搜索树中以 u 为根的子树中

结点 u 被称为这个强连通分量的根



证明

若存在结点 v 在该强连通分量中但不在以 u 为根的子树中

那么 u 到 v 的路径中肯定有一条离开子树的边

这样的边只可能是横叉边或者反祖边

然而这两类边都要求指向的结点已经被访问过了

与 u 是第一个访问的结点矛盾

Tarjan-SCC

在 Tarjan 算法中为每个结点维护了以下几个变量：

dfn_u

DFS 时结点 u 被访问的次序

low_u

能够回溯到的最早的已经在栈中的结点，设以 u 为根的子树为 subtree_u

low_u 定义为以下结点的 dfn 的最小值：

subtree_u 中的结点、从 subtree_u 通过**一条**不在搜索树上的边能到达的结点

一个结点的子树内结点的 dfn 都大于该结点的 dfn

从根开始的一条路径上的 dfn 严格递增，low 严格非降

Tarjan-SCC

按照 DFS 的次序对图中所有结点进行遍历，维护每个结点的 dfn 与 low 变量同时让遍历到的结点入栈

每当找到一个强连通分量,就按照该元素包含结点数目让栈中元素出栈

在 DFS 过程中

对于结点 u 和与其相邻的结点 v (v不是 u的父节点) 考虑 3 种情况：

- 未被访问

  继续对 v 进行 DFS，回溯时用 low_v 更新 low_u

  因存在从 u 到 v 的直接路径，所以 v 能够回溯到的已在栈中的结点 u 也一定能够回溯到

- 被访问过，在栈中

  根据 low 值的定义，用 dfn_v 更新

- 被访问过，已不在栈中

  v 已搜索完毕，其所在连通分量已被处理,不对其做操作

Tarjan-SCC

对于一个连通分量图

在该图中有且仅有一个 u 使得 $\mathrm{dfn}_u = \mathrm{low}_u$

且 u 一定是 DFS 过程中该连通分量中第一个被访问过的结点

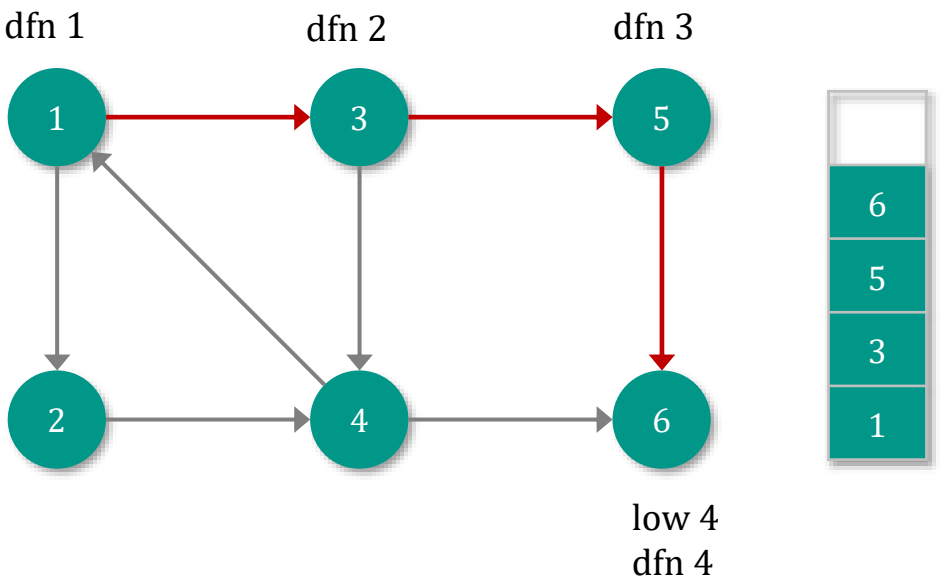
因为它的 dfn 和 low 值最小, 不会被该连通分量中的其他结点所影响

在回溯的过程中判定 $\mathrm{dfn}_u = \mathrm{low}_u$ 是否成立

若成立则栈中 u 及其上方的结点构成一个强连通分量

时间复杂度 $O(|V| + |E|)$

Tarjan-SCC

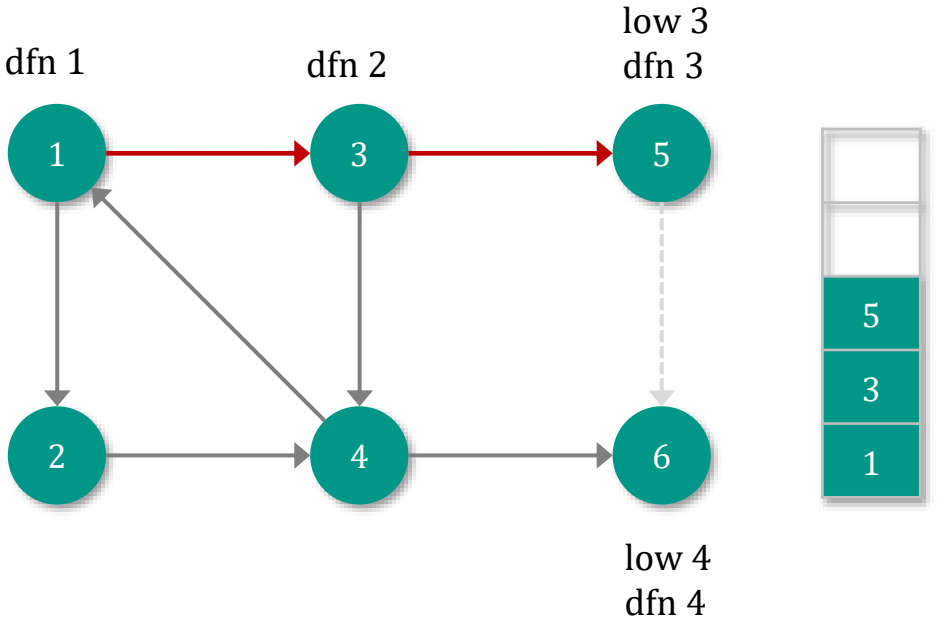


从节点 1 开始 DFS , 将遍历到的节点加入栈中

搜索到节点$u = 6$时

发现 $\text{dfn}_6 = \text{low}_6$, 找到了一个强连通分量

不断弹栈直到 $u = v$, $\{6\}$ 为一个强连通分量

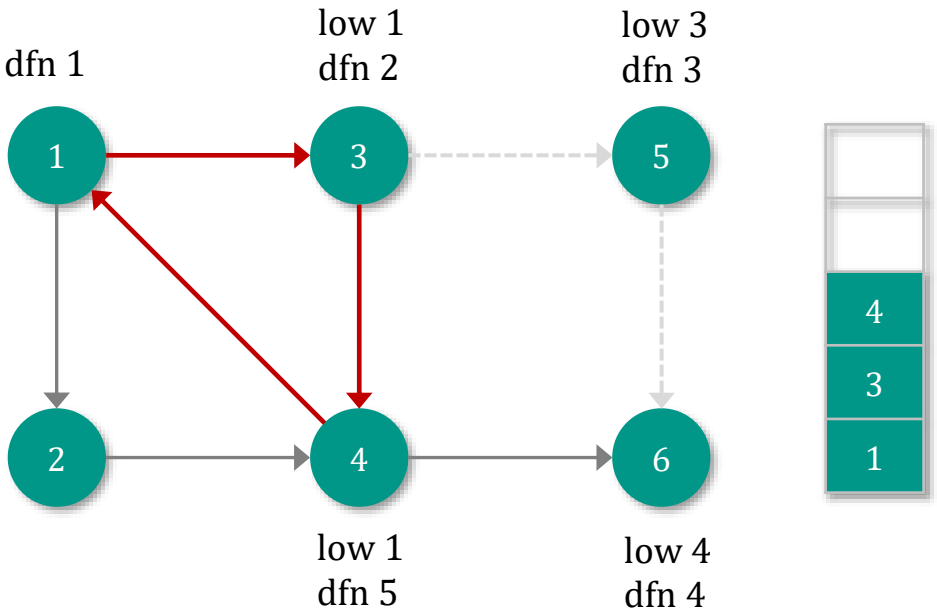


返回节点 5

发现 $\text{dfn}_5 = \text{low}_5$

不断弹栈直到 $u = v$, $\{5\}$ 为一个强连通分量

Tarjan-SCC

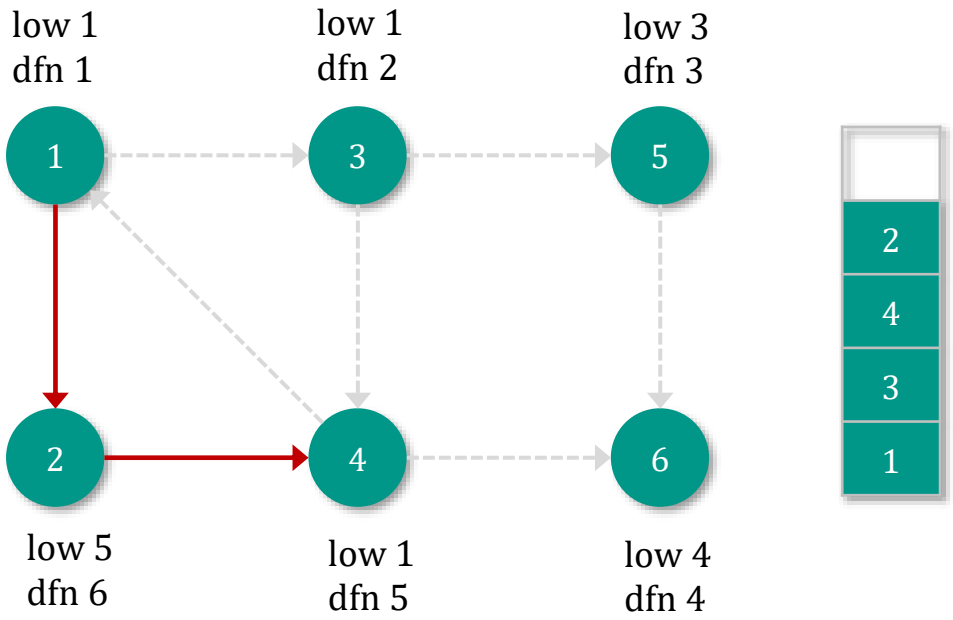


返回节点3 , 继续搜索到节点 4 , 将节点 4 加入栈

节点4 存在到节点1 的返祖边(节点1 在栈中) , 令 $\text{low}_4 = \text{dfn}_1$

节点4 存在到节点6 的横叉边(节点 6 已不在栈中)

返回节点3 , 3 $\rightarrow$ 4 为树边 , 令 $\text{low}_3 = \text{low}_4 = 1$



返回节点 1 , 最后访问节点2

节点 1 存在到节点4 的返祖边(节点4 在栈中) , 令 $\text{low}_2 = \text{dfn}_4$

返回节点 1 后发现 $\text{dfn}_1 = \text{low}_1$,找到了一个强连通分量

不断弹栈直到 $u = v$, {1,3,4,2} 为一个强连通分量

Tarjan-SCC

```
void tarjan(int u)
{
  low[u] = dfn[u] = ++idx;
  s.push(u), vis[u] = true;
  for (int i = head[u]; i; i = e[i].next)
  {
    int v = e[i].v;
    if (!dfn[v])
    {
      tarjan(v);
      low[u] = min(low[u], low[v]);
    }
    else if (vis[v])
      low[u] = min(low[u], dfn[v]);
  }
  if (low[u] == dfn[u])
  {
    cid++;
    do
    {
      cur = s.top(), s.pop();
      vis[cur] = false;
      scc[cur] = cid;
    } while (u != cur);
  }
}
```

如果 v 在栈中, 是否能够使用 low_v 来更新 low_u

即 $\text{low}_u = \min(\text{low}_u, \text{low}_v)$?

#865、有向图的强连通分量

题目描述

给定一个 n 个点 m 条边的有向图 $G = (V, E)$,每个点编号 $1 \sim n$

强连通分量是该图的极大连通子图,强连通分量的个数可能不止一个

请你找出 G 的强连通分量个数,以及最大的强连通分量点的个数

输入格式

第一输入两个正整数 n, m

接下来 m 行,每行两个正整数 u, v 表示有一条 $u \rightarrow v$

输出格式

第一行输出一个整数,强连通分量的数量

第二行输出一个整数,最大的强连通分量点数

第三行输出最大的强连通分量内的每个点(升序输出),若存在多种方案,输出字典序最小的方案

数据规模

对于全部的数据 $1 \leq u, m \leq 50000$

#896、网络传输

题目描述

在 $\mathrm{Internet}$ 网络中的每台电脑并不是直接一对一连通的,而是某些电脑之间存在单向的网络连接
也就是说存在 A 到 B 的连接不一定存在 B 到 A 的连接
并且有些连接传输速度很快,有些则很慢,所以不同连接传输所花的时间是有大有小的

另外,如果存在 A 到 B 的连接的同时也存在 B 到 A 的连接的话
那么 A 和 B 实际上处于同一局域网内,可以通过本地传输,这样花费的传输时间为 0

现在 Mas 告诉你整个网络的构成情况

他希望知道从他的电脑(编号为 1),到 $Sifo$ 的电脑(编号为 n)所需要的最短传输时间

输入格式

第一行两个整数 n, m ,表示有 n 台电脑, m 个连接关系

接下来 m 行,每行三个整数 u, v, w ,表示从电脑 u 到电脑 v 传输信息的时间为 w

> *保证 1 可以到达 n*

输出格式

输出仅一行为最短传输时间

数据规模

对于 30% 的数据, $1 \leq n \leq 10^3, 1 \leq m \leq 10^4$
对于 60% 的数据, $1 \leq n \leq 5 \times 10^3, 1 \leq m \leq 10^5$
对于 100% 的数据, $1 \leq n \leq 2 \times 10^5, 1 \leq m \leq 5 \times 10^5$

#896、网络传输

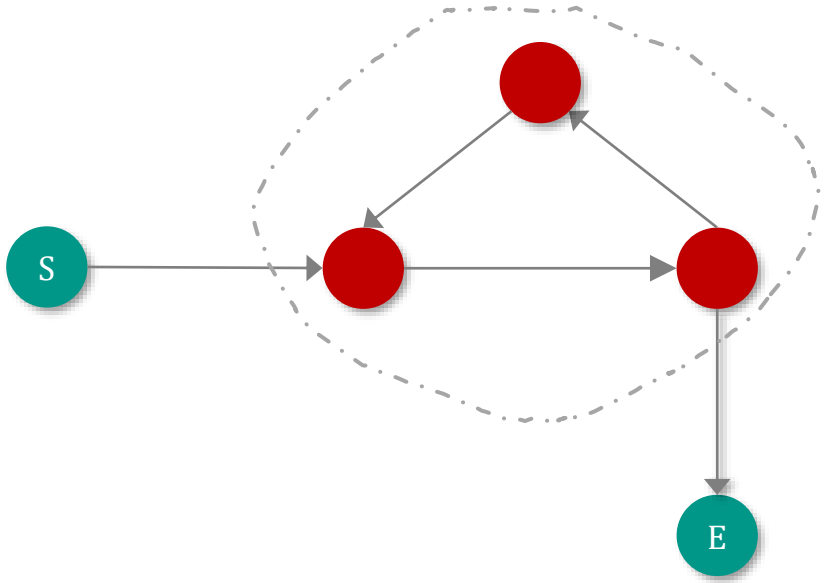
若 $u \to v, v \to u$,说明 u, v 是强连通的

局域网内节点都是强连通

一个局域网为一个强连通分量

考虑将强连通分量缩成一个点,重新建图

跑一遍 Dijkstra / SPFA




```
scanf("%d%d", &n, &m);
for (int i = 0; i < m; i++)
{
  scanf("%d%d%d", &u, &v, &w);
  edges.push_back({u, v, w});
  addEdge(u, v, w);
}
tarjan(1);
memset(head, 0, sizeof head), pos = 0; //重新建图
for (auto &&i : edges)
  if (scc[i.u] != scc[i.v]) //不在一个scc才连边
    addEdge(scc[i.u], scc[i.v], i.w);
Dijkstra(scc[1]);
printf("%d", dis[scc[n]]);
```

#686、受欢迎的牛

题目描述

每一头牛的愿望就是变成一头最受欢迎的牛

现在有 N 头牛,给你 M 对整数 (A,B),表示牛 A 认为牛 B 受欢迎

这种关系是具有传递性的,如果 A 认为 B 受欢迎, B 认为 C 受欢迎,那么牛 A 也认为牛 C 受欢迎

你的任务是求出有多少头牛被除自己之外的所有牛认为是受欢迎的

输入格式

第一行两个数 N,M

接下来 M 行,每行两个数 A,B

意思是 A 认为 B 是受欢迎的(给出的信息有可能重复,即有可能出现多个 A,B)

输出格式

输出被除自己之外的所有牛认为是受欢迎的牛的数量

数据范围

对于全部数据, $1 \leq N \leq 10^4, 1 \leq M \leq 5 \times 10^4$

将图缩点成 DAG

若 DAG 上一个点出度为 0

那么可能是受欢迎的牛，输出这个 SCC 的大小

如果有多个出度为 0 的点，那么无解

#688、网络协议

题目描述

一些学校连接在一个计算机网络上,学校之间存在软件支援协议

每个学校都有它应支援的学校名单(学校 a 支援学校 b ,并不表示学校 b 一定支援学校 a)

当某校获得一个新软件时,无论是直接得到还是网络得到,该校都应立即将这个软件通过网络传送给它应支援的学校

因此,一个新软件若想让所有连接在网络上的学校都能使用,只需将其提供给一些学校即可

请编一个程序,根据学校间支援协议(各个学校的支援名单),计算最少需要将一个新软件直接提供给多少个学校,才能使软件通过网络被传送到所有学校

如果允许在原有支援协议上添加新的支援关系,则总可以形成一个新的协议
使得此时只需将一个新软件提供给任何一个学校,其他所有学校就都可以通过网络获得该软件

编程计算最少需要添加几条新的支援关系

输入格式

第一行是一个正整数 n ,表示与网络连接的学校总数

随后 n 行分别表示每个学校要支援的学校
即: $i+1$ 行表示第 i 号学校要支援的所有学校代号,最后 0 结束

如果一个学校不支援任何其他学校,相应行则会有一个 0
一行中若有多个数字,数字之间以一个空格分隔

输出格式

包含两行,第一行是一个正整数,表示任务 a 的解
第二行也是一个正整数,表示任务 b 的解

数据规模

对于全部的数据 $2 \leq n \leq 100$

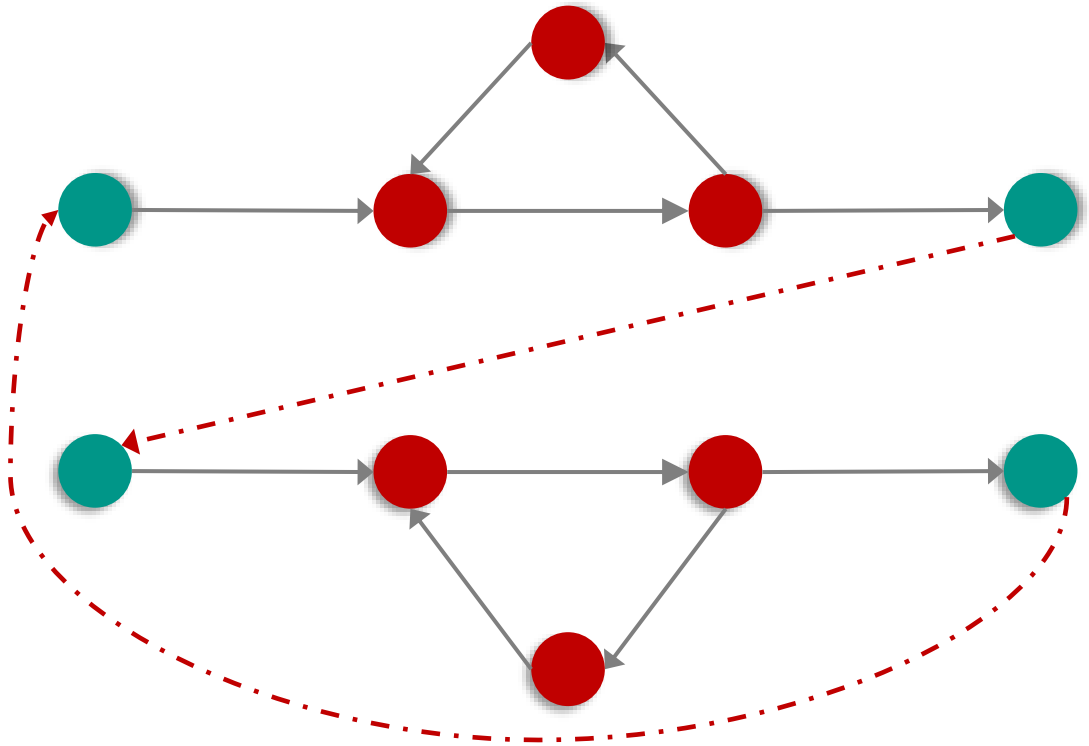
若只有一个强连通分量 ,答案显然为 1,0

#688、网络协议

考虑多个连通分量情况

记 P 为缩点后入度为 0 点的集合，Q 为缩点后出度为 0 点的集合

不妨假设假设 $|P| \leq |Q|$



- 若 $|P| = 1$

  有唯一起点且起点能到达所有点

  只需将每一个终点都向这个起点连一条边

  对于图中任意一点都可以到达所有点新加边数为 $|Q|$

- 若 $|P| \geq 2$

  即 $|Q| \geq |P| \geq 2$

  此时至少存在 2 个起点 p_1, p_2，至少存在 2 个终点 q_1, q_2，满足 p_1 能走到 q_1，p_2 能走到 q_2

  若不存在至少两个起点能走到不同的终点，则所有起点一定只能走到同一终点，而终点至少有两个

#688、网络协议

考虑从 q_1 向 p_2 连一条边，那么起点和终点的个数都会减少一个（p_2 不再是起点 q_1 不再是终点）

不断以这种方式连接 $|P|-1$ 条新边

则 $|P| \leftarrow 1$ 而 $|Q| \leftarrow |Q|-(|P|-1)$

根据前面结论，当 $|P|=1$ 时还需要再连 $|Q|-(|P|-1)$ 条新边

总添加的新边数量为 $|P|-1+|Q|-(|P|-1)=|Q|$

$|Q| \leq |P|$与上述情况对称,此时答案为 $|P|$

综上

- 若强连通分量数量为 1

  任务一答案为 1，任务二答案为 0

- 若强连通分量数量 >1

  任务一答案为 $|P|$，任务二答案为 $\max(|P|,|Q|)$

Kosaraju

Kosaraju 算法最早在 1978 年由 S. Rao Kosaraju 在一篇未发表的论文上提出，但 Micha Sharir 最早发表了它

该算法依靠两次简单的 DFS 实现

- 第一次 DFS

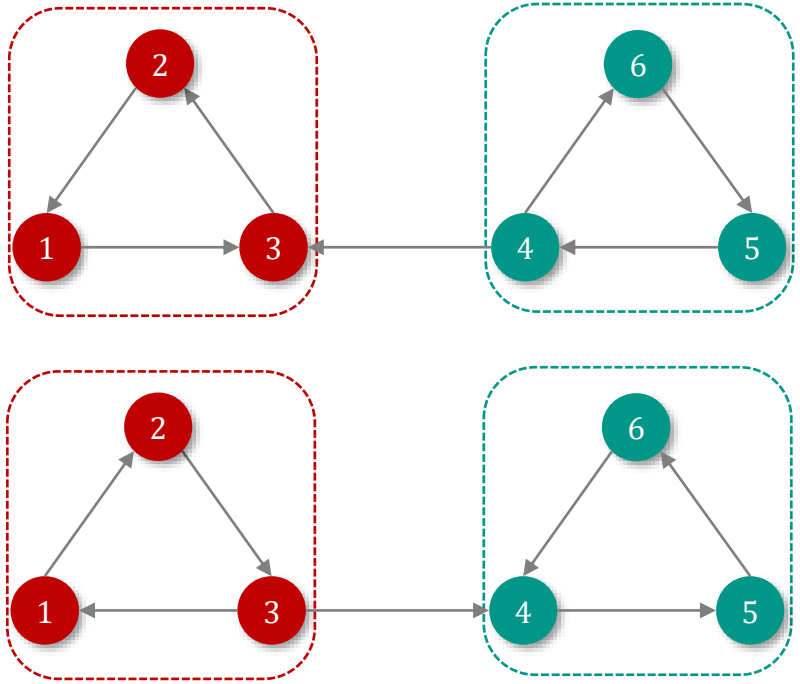
  选取任意顶点作为起点,遍历所有未访问过的顶点,并在回溯前给顶点编号

- 第二次 DFS

  考虑反图以标号最大的顶点作为起点开始 DFS

  遍历到的顶点集合就是一个强连通分量

对于所有未访问过的结点选取标号最大的,重复上述过程



两次 DFS 结束后, 强连通分量就被找出

时间复杂度为 $O(|V| + |E|)$

Kosaraju


```cpp
void dfs(int u)
{
  vis[u] = true;
  for (auto &&v : g[u])
    if (!vis[v])
      dfs(v);
  dfn[++idx] = u;
}
void dfs1(int u)
{
  vis[u] = true, scc[u] = cid, cnt[cid]++;
  for (auto &&v : rg[u])
    if (!vis[v])
      dfs1(v);
}
```

```cpp
int main()
{
  scanf("%d%d", &n, &m);
  for (int i = 0; i < m; i++)
  {
    scanf("%d%d", &u, &v);
    g[u].push_back(v), rg[v].push_back(u);
  }

  for (int i = 1; i <= n; i++)
    if (!vis[i])
      dfs(i);
  memset(vis, false, sizeof vis);
  for (int i = n; i >= 1; i--)
    if (!vis[dfn[i]])
      ++cid, dfs1(dfn[i]);

  return 0;
}
```

2-SAT

SAT 是适定性 (Satisfiability) 问题的简称

$2-\text{SAT}$ 简单的说就是给出 n 个集合，各集合有两个元素

已知若干个 $<a,b>$，表示 a 与 b 矛盾(其中 a 与 b 属于不同的集合)

从每个集合选择一个元素,判断能否选 n 个两两不矛盾的元素

注意 $2-\text{SAT}$ 问题是多项式可解的,但 $\text{k}-\text{SAT}(k>2)$ 问题是 NP 完全的

如现实场景中邀请朋友参加派对

父母二人必须去一个，而某些人间存在矛盾(如 A 先生与 B 女士有矛盾,C 女士不想和 D 先生一起出席等)

要确定能否避免来人之间没有矛盾,有时需要输出具体方案

2-SAT

若使用布尔方程表示上述问题

设 a 表示 A 先生去参加,那么 B 女士就不能参加 $\neg a$

b 表示 C 女士参加,那么 $\neg b$ 也一定成立(D 先生不参加)

即 a $\vee$ b(变量 a, b 至少满足一个)

对变量关系建有向图: $\neg a \rightarrow b \wedge \neg b \rightarrow a$ (a 不成立则 b 一定成立,同理 b 不成立则 $\neg a$ 一定成立)

对有向图跑一遍 Tarjan SCC 判断是否有一个集合中的两个元素在同一个 SCC 中，若有则输出不可能，否则输出方案

构造方案只需要把几个不矛盾的 SCC 拼合即可,输出方案时可以通过变量在图中的拓扑序确定该变量的取值

若变量 x 的拓扑序在 $\neg x$ 之后,那么取 x 值为真,即 x 所在 SCC 编号在 $\neg x$ 之前时 x 取真

因为 Tarjan 算法求强连通分量时使用了栈,所以 Tarjan 求得的 SCC 编号相当于反拓扑序

时间复杂度为 $O(n+m)$

注意 2 − SAT 只求出了任意一种方案,不能求 0 数量最少/多的方案

#3106、2-SAT

题目描述

有 N 个变量 $X_0 \sim X_{N-1}$,每个变量的可能取值为 0 或 1

给定 M 个算式,每个算式形如 $X_a\ op\ X_b = c$

其中 a, b 是变量编号, c 是数字 0 或 1 , op 是 `AND` , `OR` , `XOR` 三个位运算之一

求是否存在对每个变量的合法赋值,使所有算式都成立

输入格式

第一行包含两个整数 N 和 M

接下来 M 行,每行包含三个整数 a, b, c ,以及一个位运算(`AND` , `OR` , `XOR` 中的一个)

输出格式

输出结果
如果存在输出 `YES` ,否则输出 `NO`

数据范围

对于全部的数据 $1 \leq N \leq 1000, 1 \leq M \leq 10^6$

输入样例

```
4 4
0 1 1 AND
1 2 1 OR
3 2 0 AND
3 0 0 XOR
```

输出样例

```
YES
```

#3106、2-SAT

$i \in [0 \sim n-1]$ 表示 x_i 值为 1，$i \in [n \sim 2n-1]$ 表示 x_i 值为 0

- x_a and $x_b = 0$

  说明 x_a, x_b 至多有一个为 1

  若 $x_a = 1$ 则 $x_b = 0$ 连边 $a \to b+n$

  若 $x_b = 1$ 则 $x_a = 0$ 连边 $b \to a+n$

- x_a and $x_b = 1$

  说明 x_a, x_b 两个变量都为 1，若变量为 0 让其产生矛盾

  连边 $a+n \to a$ 和 $b+n \to b$

- x_a or $x_b = 0$

  说明 x_a, x_b 两个变量都为 0，若变量为 1 让其产生矛盾

  连边 $a \to a+n$ 和 $b \to b+n$

- x_a or $x_b = 1$

  说明 x_a, x_b 两个变量至少有一个为 1

  若 $x_a = 0$ 那么 $x_b = 1$ 连边 $a+n \to b$

  若 $x_b = 0$ 那么 $x_a = 1$ 连边 $b+n \to a$

#3106、2-SAT

- $x_a \text{ xor } x_b = 0$

  说明 x_a, x_b 两个变量相同

  若 $x_a = 0$ 那么 $x_b = 0$ 连边 $a+n \to b+n$

  若 $x_a = 1$ 那么 $x_b = 1$ 连边 $a \to b$

  若 $x_b = 0$ 那么 $x_a = 0$ 连边 $b+n \to a+n$

  若 $x_b = 1$ 那么 $x_a = 1$ 连边 $b \to a$

- $x_a \text{ xor } x_b = 1$

  说明 x_a, x_b 两个变量互不相同

  若 $x_a = 0$ 那么 $x_b = 1$ 连边 $a+n \to b$

  若 $x_a = 1$ 那么 $x_b = 0$ 连边 $a \to b+n$

  若 $x_b = 0$ 那么 $x_a = 1$ 连边 $b+n \to a$

  若 $x_b = 1$ 那么 $x_a = 0$ 连边 $b \to a+n$

跑一遍 Tarjan , 若发现 i 与 $i+n$ 在同一强连通分量内无解

原命题成立 , 逆命题是否成立? 否命题是否成立? 逆否命题是否成立?

能否使用扩展域并查集实现?

谢谢观看