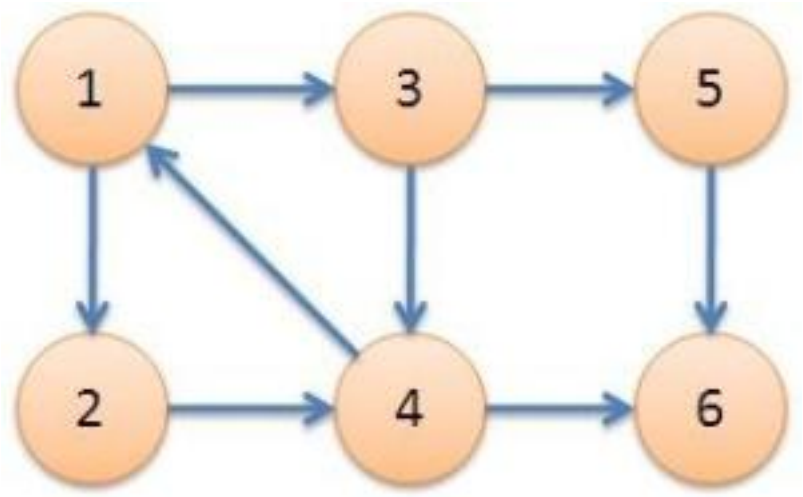
问题概述

Tarjan算法是一种由Robert·Tarjan（罗伯特·塔杨）发明的在**有向图**中求强连通分量的算法。

概念描述

如果在一个有向图中，**任意**两个顶点都可以**相互到达**，则称这个有向图为强连通图，非强连通图的极大强连通子图称为强连通分量。



例如上图中，由点1、2、3、4构成的子图就是一个强连通分量。

算法流程

首先引入两个数组dfn[]和low[]，其中dfn[i]表示点i第一次被搜索到的时间，与dfn序类似，low[i]表示在点i为根的dfs子树中，能够到达的点中dfn值的最小值。在整个算法结束后，low[]值相同的点就在同一强连通分量中。注意在初始化时low[i]=dfn[i]。

首先我们对所有dfn[i]==0的点i进行dfs，将搜到过的点压入一个栈中，如果下一个点已经在栈中，则更新low值。如果在某一个点回溯时发现这个点有dfn[x]==low[x]，说明以这个点为根的dfs树子树处于同一个强连通分量中，我们把栈顶元素弹出直到x被弹出，这些被弹出的点组成一个强连通分量。

以上面的图作为一个例子，Tarjan算法的流程如下：

dfs到1，dfn[1]=low[1]=1;

dfs到2，dfn[2]=low[2]=2;

dfs到4，dfn[4]=low[4]=3，4可以到1，1在栈中，low[4]=1;

dfs到6，dfn[6]=low[6]=4，6无法继续dfs，有dfn[6]==low[6]，从栈中弹出6，其自己作为一个强连通分量。

从4回溯到2，low[2]=1;

从2回溯到1，然后dfs到3，dfn[3]=low[3]=5，3可以到4，4在栈中，low[3]=1，

dfs到5，dfn[5]=low[5]=6，由于6已经dfs过，所以无法继续dfs，有dfn[5]==low[5]，从栈中弹出5，其自己作为一个强连通分量。

回溯到1，有dfn[1]==low[1]，从栈中弹出3、4、2、1，它们构成一个强连通分量。

例题

BZOJ1051——受欢迎的牛

题目大意

有N头牛，M对关系(a,b)表示a认为b受欢迎，如果a认为b受欢迎，b认为c受欢迎，那么a也认为c受欢迎，问有多少头牛受所有其他的牛欢迎。

解题思路

首先利用Tarjan找到强连通分量然后缩点，接下来我们会发现如果只有一个点出度为0那么就满足答案，否则结果为0。所以我们找到这个出度为0的点代表的强连通分量的大小即可。

代码实现

```
#include <map>
#include <set>
#include <ctime>
#include <cmath>
#include <stack>
#include <queue>
#include <vector>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <algorithm>
using namespace std;

const int maxn = 10005;
const int maxm = 50005;

int n, m, tot, top, scc = 0, color[maxn], dfn[maxn], low[maxn], stk[maxn],
d[maxn], siz[maxn], ins[maxn];
int u[maxm], v[maxm], ans = 0;
vector<int> g[maxn];
```

```
inline void tarjan(int x)
{
    dfn[x] = low[x] = ++tot;
    stk[++top] = x;
    ins[x] = 1;
    int s = g[x].size();
    for(int i = 0;i < s;++i)
    {
        if(!dfn[g[x][i]])
        {
            tarjan(g[x][i]);
            low[x] = min(low[x], low[g[x][i]]);
        }
        else
            low[x] = min(low[x], dfn[g[x][i]]);//这里不是low[g[x][i]]是因为g[x][i]的low值可能还没更新过
    }

    if(dfn[x] == low[x])
    {
        int now = -1;
        ++scc;
        while(now != x)
        {
            now = stk[top];
            top--;
            ins[now] = 0;
            color[now] = scc;
            ++siz[scc];
        }
    }
}

int main()
{
    scanf("%d%d", &n, &m);
    for(int i = 1;i <= m;++i)
    {
        scanf("%d%d", &u[i], &v[i]);
        g[u[i]].push_back(v[i]);
    }

    for(int i = 1;i <= n;++i)
        if(!dfn[i])
            tarjan(i);

    for(int i = 1;i <= m;++i)
        if(color[u[i]] != color[v[i]])
            ++d[color[u[i]]];
    for(int i = 1;i <= scc;++i)
    {
```

```
        if(!d[i])
        {
            if(ans > 0)
            {
                ans = 0;
                break;
            }
            ans = siz[i];
        }
    }

    printf("%d\n", ans);

    return 0;
}
```