有向图的连通性问题

对于有向图，图的连通性也可以通过搜索来完成。一个有向图是连通图，当且仅当任意两点之间至少有一条路径可以相通。非连通图的极大连通子图被称为连通分量，有向图中的连通图叫做强连通图，非强连通图中的极大连通子图被称为**强连通分量**。强连通分量另外的含义就是有向图中的环路。有向图的连通性问题大多与求强连通分量有关，本节重点介绍有向图强连通分量的求解办法。

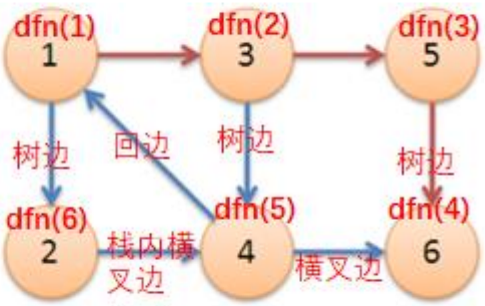
一、　Tarjan 算法

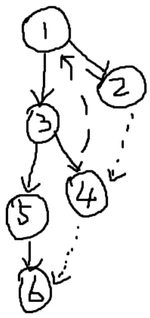
有向图的DFS与无向图的DFS的区别在于搜索只能顺边的方向进行，所以有向图的DFS不止一个根，因为从某个结点开始不一定就能走完所有的点。

另外，有向图的 DFS 除了产生**树边**和**返祖边**以外，还会有**横叉边**。我们这样定义它：

u 和 v 在已形成的 DFS 森林中没有直系上下关系，并且有 dfn[v]>dfn[u]，则称 e=uv 是**横叉边**。

注意，没有 dfn[v]<dfn[u]这种横叉边。　如下图





Tarjan 算法是基于对图深度优先搜索的算法，每个强连通分量为搜索树中的一颗子树。搜索时，把当前搜索树中未处理的节点加入一个堆栈，回溯时可以判断栈顶到栈中的节点是否为一个强连通分量。

与无向图的 Tarjan 算法一样，定义 dfn[x]为节点 x 搜索的次序编号（时间戳）。Low[x]为 x 或者 x 的子树能够追溯到的最早的栈中的节点的次序号。

Low[x]的更新由定义可以得出：

$$low[x]=\begin{cases} min\{low[x],low[y]\} & \text{边}(x,y)\text{为树边} \\ min\{low[x],dfn[x]\} & \text{边}(x,y)\text{为返祖边或栈内横叉边} \end{cases}$$

特别注意的是，对于非树边，关键需要区分返祖边与横叉边。具体实现通过是否在栈内来区分。对于非树边(x,y)，如果 y 在栈内，则表示 y 可能是 x 的祖先或是一个强连通分量的其他点，即这条边为返祖边或栈内横叉边；如果 y 不在栈内，则表示该边为栈外横叉边。

强连通的判断法则：

dfn[x]==low[x]

理解：当 dfn[x]==low[x]时表示 x 无法通过子树回到更早的祖先，那么 x 与祖先的关系只能通过父亲，这样就无法与祖先形成回路。

```
void tarjan(int u){
    stack[++index]=u;
    vis[u]=1;//入栈并标记
```

```
    LOW[u]=DFN[u]=++t_cnt;
    for(int i=head[u];i;i=E[i].next){
      int v = E[i].to;
        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]){//找到强连通
        vis[u]=0;
        scc[u]=++CN;//染色连通颜色
        size[CN]++;//染色及记录强连通分量大小
        while(u!=stack[index]){
            vis[stack[index]]=0;
            size[CN]++;//记录大小
            scc [stack[index--]]=CN;//弹栈并染色
        }
        index--;//u 出栈指针下移
    }
}
```

二、 Kosaraju 算法

该算法可用来计算有向图的强连通分量个数，并收缩强连通分量(缩点)。 算法可以说是最容易理解，最通用的算法，其比较关键的部分是同时应用了原图 G 和反图 G’。关键是记住 dfs 时的出栈顺序。

该算法具有一个隐藏性质： 如果我们把求出来的每个强连通分量收缩成一个点，并且用求出每个强连通分量的顺序来标记收缩后的结点， 那么这个顺序其实就是强连通分量收缩成点后形成的有向无环图的拓扑序列。

操作步骤如下：

1. 对原图进行 DFS 并将出栈顺序进行逆序，得到的顺序就是拓扑顺序；
2. 将原图每条边进行反向；
3. 按照①中生成顺序再进行 DFS 染色，染成同色的即一个强连通块。

证明：

设这个算法在图 G^R 中，调用 DFS(s)能够到达顶点 v，那么顶点 s 和 v 是强连通的。

两个顶点如果是强连通的，那么彼此之间都有一条路径可达，因为 DFS(s)能够达到顶点 v，因此从 s 到 v 的路径必然存在。

现在关键就是需要证明在 GR 中从 v 到 s 也是存在一条路径的，也就是要证明在 G 中存在 s 到 v 的一条路径。

而之所以 DFS(s)能够在 DFS(v)之前被调用，s 出现在 v 之前，这也就意味着，v 是在 s 之前加入该序列的。因此根据 DFS 调用的递归性质，DFS(v)应该在 DFS(s)之前返回，而有两种情形满足该条件：

1. DFS(v) START -> DFS(v) END -> DFS(s) START -> DFS(s) END

2. DFS(s) START -> DFS(v) START -> DFS(v) END -> DFS(s) END

是因为而根据目前的已知条件，GR 中存在一条 s 阿到 v 的路径，即意味着 G 中存在一条 v 到 s 的路径；

对于情况 1：调用 DFS(v)却没能在它返回前递归调用 DFS(s)，这是和 GR 中存在 s 到 s 的路径相矛盾

对于情况 2：唯一符合逻辑的调用过程

而根据 DFS(s) START -> DFS(v) START 可以推导出从 s 到 v 存在一条路径。

所以从 s 到 v 以及 v 到 s 都有路径可达，证明完毕。

三、 有向图的压缩

将有向图中的强连通子图都压缩成为一个点之后，是否和无向图压缩之后的结果一样呢？

有向图压缩之后，连接不同结点之间的边有两种：

父子边，横叉边。压缩后的图，不是一个标准意义上的树(将边看作无向)。它是一个有向无环图（DAG)，即不可再压缩的图。

压缩图参考：

设 col[x]表示节点 x 所属的强连通分量编号，以每个强连通分量的编号作为新图的节点编号建图。建图方法：

1．矩阵保存

```
for(int i=1;i<=n;i++)
    for(int j=1;j<=n;j++)
        if(g[i][j]==1&&col[i]!=col[j])
            mp[col[i]][col[j]]=1;
```

2．邻接表保存

```
for(int u=1;u<=n;u++)
    for(int k=first[u];k;k=nxt[k]){
        int v=e[k].v;
        if(col[u]==col[v])continue;
        add_new(col[u],col[v]);
    }
```

四、 实例解析

例一、P3049 上白泽慧音

题意：给定一张图，存在双向边和单向边若干，求出最大的强连通块，若有多个一样大的强连通块，则输出字典序最小的

方法一：搜索。根据方法的不同能够得到不同的分值。 时间复杂度：O(?)；期望得分：0-100

方法二：通过 Floyd 判断连通性(F[i,j]=F[i,j] or(F[i,k] and F[k,j]))，再将所有的双向连通点对统计出来，进行一次并查集。如果两个点是双向连通点对，把它们归为同一个集合。对于每个集合需要保存它的最小点编号。

时间复杂度：O(N^3)；期望得分：60

方法三：通过 Tarjan 算法求出图中所有的强连通分量，并记录每一个块的最小点编号。

最后输出最大连通分量的顶点。

时间复杂度：O(N+M)；期望得分：100

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
#define inf 1000000

int first[inf],idx=1,n,m,tot=0;
stack<int> st;
int
vis[inf]={},dfn[inf]={},low[inf]={},siz[inf]={},scc[inf]={},CN=0;
struct front_star{
    int u,v,val,nxt;
}e[inf];
void add(int u,int v,int val){
    e[++idx].u=u; e[idx].v=v; e[idx].val=val; e[idx].nxt=first[u];
first[u]=idx;
}
void tarjian(int u){
    st.push(u); vis[u]=1;
    low[u]=dfn[u]=++tot;
    for(int i=first[u];i;i=e[i].nxt){
        int v=e[i].v;
        if(!dfn[v]){
            tarjian(v);
            low[u]=min(low[u],low[v]);
        }
        else if(vis[v]) low[u]=min(low[u],dfn[v]);
    }
    if(low[u]==dfn[u]){
        ++CN;
        while(1){
            int t=st.top();st.pop();
            vis[t]=0;scc[t]=CN;siz[CN]++;
            if(t==u)break;
        }
    }
}
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++) {
        int u,v,z;
        scanf("%d%d%d",&u,&v,&z);
        add(u,v,1);
```

```
        if(z==2)
            add(v,u,1);
    }
    for(int i=1;i<=n;i++)
        if(scc[i]==0) tarjian(i);
    int ans=1,max0=0;
    for(int i=1;i<=CN;i++) if(siz[i]>siz[ans]) ans=i,max0=siz[i];
    printf("%d\n",max0);
    for(int i=1;i<=n;i++)
        if(scc[i]==ans)printf("%d ",i);
    return 0;
}
```

例二、P10091 「一本通 3.5 例 1」受欢迎的牛

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所有的牛认为是受欢迎的。

【解析】

问题抽象成图论，对于每一对欢迎关系(A,B)，我们从 A 向 B 连一条有向边，得到的图有可能存在环，在原图上利用强连通分量缩点【因为强连通子图中的任意两个点可以到达，强连通子图中所有的点具有相同的性质，即它们分别能到达的点集都是相同的，能够到达它们的点集也是相同的】，得到一个新的有向无环图，统计每一个新点的出度，对于出度为 0 的点的个数 Ans 来说：

- Ans=1：答案就是出度为 0 的强连通分量中的点数。因为其它的牛都不能被他们欢迎。
- Ans>=2：答案为 0。因为出度为 0 的点间互不欢迎

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
#define N 50010
#define M 50010
#define memt(f,k) memset(f,k,sizeof(f))
struct Edge{
    int u,v,nxt;
}e[M];

int
first[N],dfn[N],color[N],low[N],vis[N]={0},siz[N]={0},cd[N]={0};
int n,m,cnt,scc=0,tim=0;
stack<int>S;
void add(int u,int v){
    e[++cnt].u=u;e[cnt].v=v;e[cnt].nxt=first[u];first[u]=cnt;
}
void tarjan(int u){
    dfn[u]=low[u]=++tim;
    S.push(u);vis[u]=1;
```

```
    for(int i=first[u];i;i=e[i].nxt){
        int v=e[i].v;
        if(!dfn[v]){//树边
            tarjan(v);
            low[u]=min(low[u],low[v]);
        }
        else{
            if(vis[v])//回边
                low[u]=min(low[u],dfn[v]);
        }
    }
    if(dfn[u]==low[u]){
        int t;++scc;
        do{
            t=S.top();S.pop();
            vis[t]=0;
            color[t]=scc;siz[scc]++;
        }while(t!=u);
    }
}
int main(){
    cnt=0;memt(first,0);
    memt(dfn,0);memt(color,0);
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++){
        int u,v;
        scanf("%d%d",&u,&v);
        add(u,v);
    }
    for(int i=1;i<=n;i++)
        if(!dfn[i])tarjan(i);
    for(int i=1;i<=m;i++){//统计连通块的出度
        if(color[e[i].u] != color[e[i].v])
            cd[color[e[i].u]]++;
    }
    int ans=0;
    for(int i=1;i<=scc;i++){//所有连通块
        if(cd[i]==0){
            if(ans){//有超过 1 个出度为 0
                ans=0;break;
            }
            ans=i;
        }
```

```
    }
    printf("%d",siz[ans]);
    return 0;
}
```

例三、P10095 「一本通 3.5 练习 3」间谍网络

题意：一个 n 个点 m 条边的有向图，一些点有点权，表示作为起点的代价，没有点权的点无法作为起点

问题 1：是否可以从一些点出发访问完所有点

问题 2：如果可以访问所有点，最小访问代价

【解析】

对于第一问，直接枚举每个点 dfs，看是否能访问所有点。

对于第二问，如果没有环，则是入度为 0 的所有点权和

再考虑环，环上只要有点可以作为起点，所有点都可以作为起点。需要缩环处理。

得到算法：tarjan 求出每个环的最小代价，对所有环缩成一个点，新图中入度为 0 的点权和即为答案。

例四、