2020暑期专题图论一解题报告

A-**HDU1269**

**题意：**n个点，m条有向边，问是否每个点都可相互到达

**思路：**Tarjan求SCC分量是不是只有一个

**参考代码：**

#include <cstdio>

#include <iostream>

#include<queue>

#include<cstring>

#include<set>

#include<algorithm>

#include<cmath>

using namespace std;

struct node{

int to,nxt;

}e[1000000+10];

int cnt,head[10000+10];

void add(int u,int v){

e[cnt].nxt=head[u];

e[cnt].to=v;

head[u]=cnt++;

}

const int maxn=10010;

int dfn[maxn],low[maxn],scc[maxn],stk[maxn],index,sccnum,top;

void tarjan(int root){

dfn[root]=low[root]=++index;

stk[++top]=root;

for(int i=head[root];~i;i=e[i].nxt){

int v=e[i].to;

if(!dfn[v]){ //如果v结点未访问过

tarjan(v);

low[root]=min(low[root],low[v]);

}

else if(!scc[v]){ //如果还在栈内

low[root]=min(low[root],dfn[v]);

}

}

if(low[root]==dfn[root]){ //后代不能找到更浅的点

sccnum++;

for(;;){

int x=stk[top--];

scc[x]=sccnum;

if(x==root) break;

}

}

}

void init(){

cnt=0;index=0;sccnum=0;top=0;

memset(head,-1,sizeof head);

memset(scc,0,sizeof scc);

memset(dfn,0,sizeof dfn);

memset(low,0,sizeof low);

}

int main(){

int n,m;

while(~scanf("%d%d",&n,&m)){

init();

if(n==0&&m==0)break;

for(int i=0;i<m;i++){

int x,y;

scanf("%d%d",&x,&y);

add(x,y);

}

for(int i=1;i<=n;i++){

if(!scc[i])tarjan(i);

}

if(sccnum==1)printf("Yes\n");

else printf("No\n");

}

return 0;

}

## **B-HDU5934**

**题意：**有N个炸弹，每个炸弹有一个坐标，一个爆炸范围和一个爆炸花费，如果一个炸弹的爆炸范围内有另外的炸弹，那么如果该炸弹爆炸，就会引爆所有爆炸范围内的炸弹，求让所有炸弹爆炸的最小花费。

**思路：**先n^2的把每个炸弹爆炸范围内的炸弹都连一条有向边，然后再找强连通分量缩点，这样会形成多个有向无环图，然后对于每个有向无环图找一个入度为0的点，找这个入度为0的点里面耗费最小的炸弹去引爆，就可以了。

**参考代码：**

#include <cstdio>

#include <iostream>

#include<queue>

#include<cstring>

#include<set>

#include<algorithm>

using namespace std;

typedef long long ll;

const int maxn=1010;

const int maxm=1000010;

struct node{

int to,nxt;

}e[maxm\*4];

int head[maxn],cnt;

void add(int u,int v){

e[cnt].to=v;

e[cnt].nxt=head[u];

head[u]=cnt++;

}

int dfn[maxn],low[maxn],scc[maxn],stk[maxn],index=0,sccnum=0,top=0;

ll cost[maxn];

int rd[maxn];

struct Point{

ll x,y,r,c;

}p[maxn];

void tarjan(int root){

dfn[root]=low[root]=++index;

stk[++top]=root;

for(int i=head[root];~i;i=e[i].nxt){

int v=e[i].to;

if(!dfn[v]){ //如果v结点未访问过

tarjan(v);

low[root]=min(low[root],low[v]);

}

else if(!scc[v]){ //如果还在栈内

low[root]=min(low[root],dfn[v]);

}

}

if(low[root]==dfn[root]){ //后代不能找到更浅的点

sccnum++;

for(;;){

int x=stk[top--];

scc[x]=sccnum;

cost[sccnum]=min(cost[sccnum],p[x].c);

if(x==root) break;

}

}

}

void init(){

sccnum=0;

index=0;

top=0;

cnt=0;

memset(dfn,0,sizeof dfn);

memset(low,0,sizeof low);

memset(scc,0,sizeof scc);

memset(head,-1,sizeof head);

memset(rd,0,sizeof rd);

for(int i=0;i<=maxn;i++)cost[i]=1e9;

}

int main()

{

int tcas;

scanf("%d",&tcas);

for(int t=1;t<=tcas;t++){

int n;

scanf("%d",&n);

init();

for(int i=1;i<=n;i++)

scanf("%lld%lld%lld%lld",&p[i].x,&p[i].y,&p[i].r,&p[i].c);

for(int i=1;i<=n;i++){

for(int j=i+1;j<=n;j++){

ll dis=(p[i].x-p[j].x)\*(p[i].x-p[j].x)+(p[i].y-p[j].y)\*(p[i].y-p[j].y);

if(dis<=p[i].r\*p[i].r)add(i,j);

if(dis<=p[j].r\*p[j].r)add(j,i);

}

}

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

for(int i=1;i<=n;i++){

for(int j=head[i];~j;j=e[j].nxt){

if(scc[i]!=scc[e[j].to]){

rd[scc[e[j].to]]++;

}

}

}

ll ans=0;

for(int i=1;i<=sccnum;i++)if(rd[i]==0)ans+=cost[i];

printf("Case #%d: %lld\n",t,ans);

}

return 0;

}

## **C-HDU3394**

**题意：**有一个公园有n个景点，公园的管理员准备修建m条道路，并且安排一些形成回路的参观路线。如果一条道路被多条道路公用，那么这条路是冲突的；如果一条道路没在任何一个回路内，那么这条路是不冲突的。问分别有多少条有冲突的路和没有冲突的路

**思路：**首先非多余的边很明显是桥，因为其不在任何一个回路中。

那么冲突边是多个回路共用的边，注意此处的回路是简单回路，那么考虑点BCC缩点建图，如果一个点BCC里的边数大于点数，那么其至少2个环，其中的所有边的都是冲突边。如果点数等于边数，那么只有一个大环，没有冲突变，而点BCC中不存在点数少于边数的情况。

**参考代码：**

#include <cstdio>

#include <iostream>

#include<cstring>

#include<algorithm>

using namespace std;

typedef long long ll;

const int maxn=10010;

const int maxm=100010;

struct node{

int v,nxt;

}edge[maxm<<1];

int head[maxn],cnt;

void add(int u,int v){

edge[cnt].nxt=head[u];

edge[cnt].v=v;

head[u]=cnt++;

}

int dfn[maxn],low[maxn],idx,ef[maxm<<1],st[maxn],belong[maxm<<1],

isCut[maxm<<1],top,bccnum,ans1,ans2,bj[maxn];

void tarjan(int root,int fa){

dfn[root]=low[root]=++idx;

//新点初始化

int child=0;

//初始节点需要两个以上儿子且dfn[root]<=low[v]才是割点

for(int i=head[root];~i;i=edge[i].nxt){

int v=edge[i].v;

if(ef[i])continue;

ef[i]=ef[i^1]=1;

st[++top]=i;//边入栈

if(!dfn[v]){

child++;

tarjan(v,root);

low[root]=min(low[root],low[v]);

if(dfn[root]<low[v])ans1++;//判桥

if(dfn[root]<=low[v]){

isCut[root]=1;

bccnum++;

int num1=0,num2=0;//记录点bcc中边的数量和点的数量

for(;;){

num1++;

int j=st[top--];

if(bj[edge[j].v]!=bccnum){bj[edge[j].v]=bccnum;num2++;}

if(bj[edge[j^1].v]!=bccnum){bj[edge[j^1].v]=bccnum;num2++;}

belong[(j>>1)+1]=bccnum;//标记边所属的bcc

if(i==j)break;

}

if(num1>num2)ans2+=num1;

}

}

else low[root]=min(low[root],dfn[v]);

}

if(root==fa&&child<2)isCut[root]=0;

//如果初始节点没有两个以上的儿子，标记清零 isCut[i]=1表示该点是割点

}

void init(){

memset(head,-1,sizeof head);

memset(dfn,0,sizeof dfn);

memset(ef,0,sizeof ef);

memset(belong,0,sizeof belong);

memset(bj,0,sizeof bj);

cnt=idx=top=bccnum=ans1=0;ans2=0;

}

int main(){

int n,m;

while(~scanf("%d%d",&n,&m)){

if(n==0&&m==0)break;

init();

for(int i=0;i<m;i++){

int x,y;

scanf("%d%d",&x,&y);

add(x,y);

add(y,x);

}

for(int i=0;i<n;i++) if(!dfn[i])tarjan(i,-1);

printf("%d %d\n",ans1,ans2);

}

return 0;

## **D-HDU4966**

**题意：**总共有n门课，每门也有一个最高等级ai，一开始每门都是等级0。接着有m个提升班，每个班级有a l1 b l2 w，表示只有第a门课程在l1及以上时才能上，花费代价w，可以讲第b门课程提升到l2等级。现在就将所有的课程提升到最高等级需要付出的最小代价，若无法达到则输出-1。

**思路：**所有课程的每个等级视为一个节点，对于每门课程的等级i,可以建一条对等级i-1的有向边，边权为0；对于每个提升班，可以建一条边权为w的从点(a,l1)到(b,l2)的有向边。那么根据题意，就是求这幅图的最小生成树。最小生成树的所有边的权值之和就是答案了。

**参考代码：**

#include <cstdio>

#include <iostream>

#include<queue>

#include<cstring>

#include<set>

#include<algorithm>

using namespace std;

struct node{

int from,to,nxt,val;

}ed[500000+10];

int head[100000+10],cnt;

void add(int u,int v,int w){

ed[cnt].from=u;

ed[cnt].to=v;

ed[cnt].val=w;

ed[cnt].nxt=head[u];

head[u]=cnt++;

}

int in[100000+10],pre[100000+10],rt,vis[100000+10],idx[100000+10];

const int inf=1e9;

int Zhuliu(int root,int totn,int totm){

int x,y,ans=0;

while(true){

for(int i=0;i<=totn;i++)in[i]=inf;

for(int i=0;i<totm;i++){

node& e=ed[i];

if(e.from!=e.to&&e.val<in[e.to]) {

in[e.to]=e.val;

pre[e.to]=e.from;

if(e.from==root) rt=i;

}

}

for(int i=0;i<=totn;i++) if(in[i]==inf&&i!=root)return -1;

//如有孤立的点一定无法构成最小生成树。

int tot=0;

memset(vis,-1,sizeof vis);

memset(idx,-1,sizeof idx);

in[root]=0;

for(int i=0;i<=totn;i++){

ans+=in[i];

y=i;

while(vis[y]!=i&&idx[y]==-1&&y!=root){

vis[y]=i;

y=pre[y];

}

//找环。

while(y!=root&&idx[y]==-1){

for(x=pre[y];x!=y;x=pre[x]) idx[x]=tot;

idx[y]=tot++;

}

//对环进行收缩。

}

if(!tot) break;

//如果没有找到新的缩点就结束算法。

for(int i=0;i<=totn;i++) if(idx[i]==-1) idx[i]=tot++;

for(int i=0;i<totm;i++){

int y=ed[i].to;

ed[i].from=idx[ed[i].from];

ed[i].to =idx[ed[i].to];

if(ed[i].from!=ed[i].to) ed[i].val-=in[y];

}

//新建图。

totn=tot-1;

root=idx[root];

//保持根不变

}

return ans;

}

int level[55],sub[55];

void init(){

memset(head,-1,sizeof head);

memset(sub,0,sizeof sub);

cnt=0;

}

int main(){

int n,m;

while(~scanf("%d%d",&n,&m)){

if(n==0&&m==0)break;

init();

for(int i=1;i<=n;i++){

scanf("%d",&level[i]);

level[i]++;

sub[i]=sub[i-1]+level[i];

add(0,sub[i-1]+1,0);//0为虚根，在0与每门课的0等级间建一条权值为0的边

for(int j=1;j<level[i];j++){

add(j+sub[i-1]+1,j+sub[i-1],0);//对于每一门课，在j+1等级到j等级间建一条权值为0的边

}

}

for(int i=0;i<m;i++){

int a,ax,b,bx,w;

scanf("%d%d%d%d%d",&a,&ax,&b,&bx,&w);

add(sub[a-1]+ax+1,sub[b-1]+bx+1,w);

}

int ans=Zhuliu(0,sub[n],cnt);

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

}

}

## **E-HDU2121**

**题意：**有n个城市，问能否在这n个城市中选一个做首都 , 要求首都都能到其他城市 , 道路花费要最少 , 且道路都是单向的。

**思路：**添加一个虚根，在这个虚根和每个结点间连一条边，边的权值大于原图中所有边权值和 ，以虚根为根跑最小树形图，如果能得到ans，并且ans-sum>sum，我们就可以认为通过这个虚拟节点我们连出去两条以上的边，即原图是不连通的，我们就可以认为不存在最小树形图。反之原图必联通，且权值和为ans - sum。那么真正的根呢，在找最小入弧时，如果这条弧的起点是虚拟根，那么这条弧的终点就是要求的根，因为如果有多解的话，必然存在一个环，环上的顶点都可以做根，但是我们根据最小入边的性质，可知，如果没缩点，必然找不到那个根，因为虚拟根连的边都非常大。但是缩点后，找到的必然是最小的那个序号的根

**参考代码：**

#include <cstdio>

#include <iostream>

#include<queue>

#include<cstring>

#include<set>

#include<algorithm>

using namespace std;

typedef long long ll;

const int maxn=1010;

const int maxm=10010;

struct node{

int from,to,nxt,val;

}ed[maxm\*2];

ll in[maxn];

int pre[maxn],rt,vis[maxn],idx[maxn];

const ll inf=1e17;

ll Zhuliu(int root,int totn,int totm){

int x,y;

ll ans=0;

while(true){

for(int i=0;i<=totn;i++)in[i]=inf;

for(int i=0;i<totm;i++){

node& e=ed[i];

if(e.from!=e.to&&e.val<in[e.to]) {

in[e.to]=e.val;

pre[e.to]=e.from;

if(e.from==root) rt=i;

}

}

for(int i=0;i<=totn;i++) if(in[i]==inf&&i!=root)return -1;

//如有孤立的点一定无法构成最小生成树。

int tot=0;

memset(vis,-1,sizeof vis);

memset(idx,-1,sizeof idx);

in[root]=0;

for(int i=0;i<=totn;i++){

ans+=in[i];

y=i;

while(vis[y]!=i&&idx[y]==-1&&y!=root){

vis[y]=i;

y=pre[y];

}

//找环。

while(y!=root&&idx[y]==-1){

for(x=pre[y];x!=y;x=pre[x]) idx[x]=tot;

idx[y]=tot++;

}

//对环进行收缩。

}

if(!tot) break;

//如果没有找到新的缩点就结束算法。

for(int i=0;i<=totn;i++) if(idx[i]==-1) idx[i]=tot++;

for(int i=0;i<totm;i++){

int y=ed[i].to;

ed[i].from=idx[ed[i].from];

ed[i].to =idx[ed[i].to];

if(ed[i].from!=ed[i].to) ed[i].val-=in[y];

}

//新建图。

totn=tot-1;

root=idx[root];

//保持根不变

}

return ans;

}

int main(){

int n,m;

while(~scanf("%d%d",&n,&m)){

ll sum=1;

for(int i=0;i<m;i++){

scanf("%d%d%d",&ed[i].from,&ed[i].to,&ed[i].val);

ed[i].from++;

ed[i].to++;

sum+=ed[i].val;

}

for(int i=1;i<=n;i++){

ed[i+m-1].from=0;

ed[i+m-1].to=i;

ed[i+m-1].val=sum;

}

ll ans=Zhuliu(0,n,m+n);

if(ans-sum>=sum||ans==-1)printf("impossible\n\n");

else printf("%lld %d\n\n",ans-sum,rt-m);

}

return 0;

}

## **F-POJ1236**

**题意：**

给定一个有向图，N个点，求：

1)至少要选几个顶点，才能做到从这些顶点出发，可以到达全部顶点

2)至少要加多少条边，才能使得从任何一个顶点出发，都能到达全部顶点

**思路**：

Tarjan算法求SCC，并缩点建图。

那么对于问题1，新的图中入度为0点的点数即是答案。

对于问题2，答案为max（入度为0的点的点数，出度为0的点数），因为对于每个入度或出度为0的点，需要连一条边来解决，那么将出度为0的点连向入度为0的点是最优的。

此外，如果最后SCC只有1个，那么问题2的答案应该特判为0。

**参考代码：**

#include <cstdio>

#include <iostream>

#include<queue>

#include<cstring>

#include<set>

#include<algorithm>

using namespace std;

typedef long long ll;

struct node{

int to,nxt;

}edge[400000+10];

const int maxn=110;

int head[maxn],cnt;

void add(int u,int v){

edge[cnt].to=v;

edge[cnt].nxt=head[u];

head[u]=cnt++;

}

int dfn[maxn],low[maxn],scc[maxn],stck[maxn],top,sccnum,index;

void tarjan(int rt){

dfn[rt]=low[rt]=++index;

stck[++top]=rt;

for(int i=head[rt];~i;i=edge[i].nxt){

int to=edge[i].to;

if(!dfn[to]){

tarjan(to);

low[rt]=min(low[rt],low[to]);

}

else if(!scc[to]){

low[rt]=min(low[rt],dfn[to]);

}

}

if(low[rt]==dfn[rt]){

sccnum++;

while(1){

int x=stck[top--];

scc[x]=sccnum;

if(x==rt)break;

}

}

}

int rd[110],cd[110];

void init(){

sccnum=0;

index=0;

top=0;

cnt=0;

memset(dfn,0,sizeof dfn);

memset(low,0,sizeof low);

memset(scc,0,sizeof scc);

memset(head,-1,sizeof head);

memset(rd,0,sizeof rd);

memset(cd,0,sizeof cd);

}

int main()

{

int n;

scanf("%d",&n);

init();

for(int i=1;i<=n;i++){

while(1){

int x;

scanf("%d",&x);

if(x==0)break;

add(i,x);

}

}

for(int i=1;i<=n;i++){

if(!scc[i])tarjan(i);

}

for(int i=1;i<=n;i++){

for(int j=head[i];~j;j=edge[j].nxt){

if(scc[i]!=scc[edge[j].to]){

rd[scc[edge[j].to]]++;

cd[scc[i]]++;

}

}

}

int ans,cnum=0,rnum=0;

for(int i=1;i<=sccnum;i++){

if(rd[i]==0)rnum++;

if(cd[i]==0)cnum++;

}

if(sccnum==1)printf("1\n0\n");

else ans=max(rnum,cnum);

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

return 0;

}

## **G-POJ3177**

**题意：**有n个牧场，Bessie 要从一个牧场到另一个牧场，要求至少要有2条独立的路可以走。现已有m条路，求至少要新建多少条路，使得任何两个牧场之间至少有两条独立的路。两条独立的路是指：没有公共边的路，但可以经过同一个中间顶点。给出的图保证已经联通。

**思路：**在同一个边双连通分量中，任意两点都有至少两条独立路可达，所以同一个边双连通分量里的所有点可以看做同一个点。缩点后，新图是一棵树，因为原图已联通。现在就是要在树上添边，令所有点缩为一个边双联通分量。

答案就是(树上度为1的点数+1）/2。

**参考代码：**

#include <cstdio>

#include <iostream>

#include<queue>

#include<cstring>

#include<set>

#include<algorithm>

using namespace std;

typedef long long ll;

const int maxn=5010;

const int maxm=10010;

struct node{

int to,nxt;

}edge[maxm\*4];

int head[maxn],cnt;

void add(int u,int v){

edge[cnt].to=v;

edge[cnt].nxt=head[u];

head[u]=cnt++;

}

int dfn[maxn],low[maxn],bcc[maxn],st[maxn],top,bccnum,index,ef[maxm\*4];

void tarjan(int root){

dfn[root]=low[root]=++index;//新点初始化

st[++top]=root; //节点入栈

for(int i=head[root];~i;i=edge[i].nxt){//遍历root指出去的边

int v=edge[i].to;

if(ef[i])continue;

ef[i]=ef[i^1]=1; //标记走过的边

if(!dfn[v]){//如果v节点未去过，搜索v节点

tarjan(v);

low[root]=min(low[root],low[v]);//更新low值

}

else low[root]=min(low[root],dfn[v]);

//与有向图区分，此处else不需要判别v节点是否在栈内

}

if(low[root]==dfn[root]){

bccnum++;//得到一个BCC

for(;;){

int x=st[top--];//节点出栈，并标记其BCC

bcc[x]=bccnum;

if(x==root)break;

}

}

}

int rd[maxn];

void init(){

bccnum=0;index=0;top=0;cnt=0;

memset(dfn,0,sizeof dfn);

memset(low,0,sizeof low);

memset(bcc,0,sizeof bcc);

memset(head,-1,sizeof head);

memset(rd,0,sizeof rd);

memset(ef,0,sizeof ef);

}

int main()

{

int n,m,ans=0;

scanf("%d%d",&n,&m);

init();

for(int i=1;i<=m;i++){

int x,y;

scanf("%d%d",&x,&y);

add(x,y);add(y,x);

}

for(int i=1;i<=n;i++){

if(!bcc[i])tarjan(i);

}

for(int i=1;i<=n;i++){

for(int j=head[i];~j;j=edge[j].nxt){

if(bcc[i]!=bcc[edge[j].to]){

rd[bcc[edge[j].to]]++;

}

}

}

for(int i=1;i<=bccnum;i++){

if(rd[i]==1)ans++;

}

printf("%d\n",(ans+1)/2);

return 0;

## }

## **H-POJ3164**

**题意：**n个被破坏的网络节点，可以在一些结点间连接单向电线，能使每个节点都能收到从1号结点发出的指令最少需要多少电线。如果不能输出“poor snoopy”。

**思路：**最小树形图模板题

**参考代码：**

#include <cstdio>

#include <iostream>

#include<queue>

#include<cstring>

#include<set>

#include<algorithm>

#include<cmath>

using namespace std;

struct node{

int from,to,nxt;

double val;

}ed[500000+10];

int head[100000+10],cnt;

const int maxn=20000;

void add(int u,int v,double w){

ed[cnt].from=u;

ed[cnt].to=v;

ed[cnt].val=w;

ed[cnt].nxt=head[u];

head[u]=cnt++;

}

double in[maxn];

const double inf=1e9;

int pre[maxn],vis[maxn],idx[maxn];

double Zhuliu(int root,int totn,int totm){//totn:点的编号0~totn totm:边的编号0~(totm-1)

int x,y;

double ans=0;

while(true){

for(int i=0;i<=totn;i++)in[i]=inf;

for(int i=0;i<totm;i++){

node& e=ed[i];

if(e.from!=e.to&&e.val<in[e.to]) {

in[e.to]=e.val;

pre[e.to]=e.from;

}

}

for(int i=0;i<=totn;i++) if(in[i]>=inf&&i!=root)return -1;

//如有孤立的点一定无法构成最小生成树。

int tot=0;

memset(vis,-1,sizeof vis);

memset(idx,-1,sizeof idx);

in[root]=0;

for(int i=0;i<=totn;i++){

ans+=in[i];

y=i;

while(vis[y]!=i&&idx[y]==-1&&y!=root){

vis[y]=i;

y=pre[y];

}

//找环。

while(y!=root&&idx[y]==-1){

for(x=pre[y];x!=y;x=pre[x]) idx[x]=tot;

idx[y]=tot++;

}

//对环进行收缩。

}

if(!tot) break;

//如果没有找到新的缩点就结束算法。

for(int i=0;i<=totn;i++) if(idx[i]==-1) idx[i]=tot++;

for(int i=0;i<totm;i++){

int y=ed[i].to;

ed[i].from=idx[ed[i].from];

ed[i].to =idx[ed[i].to];

if(ed[i].from!=ed[i].to) ed[i].val-=in[y];

}

//新建图。

totn=tot-1;

root=idx[root];

//保持根不变

}

return ans;

}

void init(){

memset(head,-1,sizeof head);

cnt=0;

}

struct Point{

double x,y;

}p[210];

double getdis(Point a,Point b){

return sqrt((a.x-b.x)\*(a.x-b.x)+(a.y-b.y)\*(a.y-b.y));

}

int main(){

int n,m;

while(~scanf("%d%d",&n,&m)){

init();

for(int i=0;i<n;i++){

scanf("%lf%lf",&p[i].x,&p[i].y);

}

for(int i=0;i<m;i++){

int x,y;

scanf("%d%d",&x,&y);

add(x-1,y-1,getdis(p[x-1],p[y-1]));

}

double ans=Zhuliu(0,n-1,cnt);

if(ans<0)printf("poor snoopy\n");

else printf("%.2f\n",ans);

}

}

1. **HDU6604 Blow up the city**

****题意****：给你一个DAG，定义中心城市为没有出度的点。一个DAG里会存在多个中心城市。q次询问x,y。问炸掉一个城市使得a,b至少有一个不能到达中心城市得方案数。

题解: 反向建图。由于可能有多个中心城市，需要定义一个超级起点到所有入度为0的点，然后建支配树，每次询问的答案就是: depth[a]+depth[b]-depth[lca(a,b)], (超级起点的depth是0)

代码：

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;typedef double db;

typedef pair<int, int> pii;typedef pair<ll, ll> pll;

typedef pair<int,ll> pil;typedef pair<ll,int> pli;

#define Fi first

#define Se second

#define \_Out(a) cerr<<#a<<" = "<<(a)<<endl

const int INF = 0x3f3f3f3f, MAXN = 2e5 + 50;

const ll LINF = 0x3f3f3f3f3f3f3f3f, MOD = 1004535809;

struct Graph

{

vector<int>to[MAXN];

void init(int n){for(int i=1;i<=n;++i)to[i].clear();}

void addE(int a,int b){to[a].push\_back(b);}

};

namespace ZPT

{

bool mt;

Graph rG,DAG\_zpt;

int rt,n;

int tim,dfn[MAXN],idx[MAXN],fa[MAXN],mndfn[MAXN],semi[MAXN],dsu[MAXN],idom[MAXN];

int fifa(int x){

if(x==dsu[x])return x;

int nowfa=dsu[x];dsu[x]=fifa(dsu[x]);

if(dfn[semi[mndfn[nowfa]]]<dfn[semi[mndfn[x]]])mndfn[x]=mndfn[nowfa];

return dsu[x];

}

void find\_dfn(Graph &G,int u,int las)

{

dfn[u]=++tim;idx[tim]=u;fa[u]=las;

for(int i=0;i<G.to[u].size();i++)

{

int v=G.to[u][i];

if(v==las)continue;

if(!dfn[v])find\_dfn(G,v,u);

}

}

void buildQAQ(Graph &G,Graph &zpt){

find\_dfn(G,rt,-1);

for(int nowdfn=n;nowdfn>=2;nowdfn--){

int u=idx[nowdfn];

for(int i=0;i<rG.to[u].size();i++){

int v=rG.to[u][i];

if(!dfn[v])continue;

fifa(v);

if(dfn[semi[mndfn[v]]]<dfn[semi[u]])semi[u]=semi[mndfn[v]];

}

dsu[u]=fa[u];

DAG\_zpt.addE(semi[u],u);

u=fa[u];

for(int i=0;i<DAG\_zpt.to[u].size();i++)

{

int v=DAG\_zpt.to[u][i];fifa(v);

if(semi[mndfn[v]]==u)idom[v]=u;//u==semi[v]

else idom[v]=mndfn[v];

}

}

for(int nowdfn=2;nowdfn<=n;nowdfn++){

int u=idx[nowdfn];

if(idom[u]!=semi[u])idom[u]=idom[idom[u]];

}

for(int i=1;i<=n;i++)if(i!=rt)zpt.addE(idom[i],i);

}

void build(Graph &origin\_Graph,Graph &ret,int node\_number,int root)//node\_number: node numbered from 1 to n, root: the node where you start

{

n=node\_number;

for(int i=1;i<=n;i++)dsu[i]=semi[i]=mndfn[i]=i,dfn[i]=0;

if(mt)rG.init(n),DAG\_zpt.init(n),ret.init(n);

mt=1;rt=root;tim=0;

for(int i=1;i<=n;i++)for(int j=0;j<origin\_Graph.to[i].size();j++)rG.addE(origin\_Graph.to[i][j],i);

buildQAQ(origin\_Graph,ret);

}

}

int n,m;

Graph G,zpt;

int depth[MAXN],grand[MAXN][20];

void LCA\_Dfs(int u, int fa,int dep)

{

depth[u] = dep; grand[u][0] = fa;

for (auto v:zpt.to[u])if(v != fa)LCA\_Dfs(v, u, dep + 1);

}

inline void LCA\_Init()

{

LCA\_Dfs(n+1, -1, 0);

for (int i = 1; i <= 18; i++)for (int j = 1; j <= 1+n; j++)grand[j][i] = (grand[j][i - 1] == -1 ? -1 : grand[grand[j][i - 1]][i - 1]);

}

inline int Find\_LCA(int a, int b)

{

if (depth[a] > depth[b])swap(a, b);

for (int i = 18; i >= 0; --i) if (grand[b][i] != -1 && depth[grand[b][i]] >= depth[a])b = grand[b][i];

if (a == b)return a;

for (int i = 18; i >= 0; --i)if (grand[b][i] != grand[a][i])a = grand[a][i], b = grand[b][i];

return grand[a][0];

}

bool isin[MAXN];

void work()

{

int T;

scanf("%d",&T);

while(T--)

{

scanf("%d%d",&n,&m);

G.init(n+1);

for(int i=1;i<=n+1;i++)isin[i]=0;

for(int i=1;i<=m;i++)

{

int u,v;scanf("%d%d",&u,&v);

G.addE(v,u);

isin[u]=1;

}

for(int i=1;i<=n;i++)if(!isin[i])G.addE(n+1,i);

ZPT::build(G,zpt,n+1,n+1);

LCA\_Init();

int q;scanf("%d",&q);

for(int cas=1;cas<=q;cas++)

{

int a,b;scanf("%d%d",&a,&b);

int lca=Find\_LCA(a,b);

printf("%d\n",depth[a]+depth[b]-depth[lca(a,b)]);

}

}

}

int main(){

work();

}