// Tarjan求割点

// vector<int> ans中存储了所有边双连通分量的代表元

void tarjan(int cur,int father)

{

visited[cur]=true;

dfn[cur]=low[cur]=++ind;

int i,to,chil=0;

bool f=false;

for (i=0;i<v[cur].size();i++)

{

to=v[cur][i];

if (!visited[to])

{

tarjan(to,cur);chil++;

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

if (father==-1 && chil>1 && !f) {ans.pb(cur);f=true;}

//根节点有多个儿子肯定是割点

if (father!=-1 && low[to]>=dfn[cur] && !f) {ans.pb(cur);f=true;}

}

else if (to!=father) low[cur]=min(low[cur],low[to]);

}

}

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

// Tarjan求强连通分量+缩点

const int MAXN=1e5;

vector<int> v[MAXN+48];

int dfn[MAXN+48],low[MAXN+48],ind=0,tot;

stack<int> s;

void tarjan(int cur)

{

int i,to;

visited[cur]=true;

dfn[cur]=low[cur]=++ind;

s.push(cur);

for (i=0;i<v[cur].size();i++)

{

to=v[cur][i];

if (!dfn[to])

{

tarjan(to);

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

}

else if (visited[to]) low[cur]=min(low[cur],low[to]);

}

if (dfn[cur]==low[cur])

{

tot++;

int v1;

do

{

v1=s.top();

anc[v1]=tot;

w[tot]+=w[v1];

// 可能需要统计SCC的权值和etc.

visited[v1]=false;

s.pop();

}

while (v1!=cur);

}

}

Int main ()

{

tot=n;

//从n+1开始给强连通分量编号

for (i=1;i<=n;i++) if (!a[i].ind) tarjan(i);

}

// Tarjan求边双连通分量

// 基本思想：求出所有的桥，然后做一遍dfs即可

void tarjan(int cur,int father)

{

dfn[cur]=low[cur]=++ind;

int i,to;

for (i=0;i<v[cur].size();i++)

{

to=edge[v[cur][i]].x==cur?edge[v[cur][i]].y:edge[v[cur][i]].x;

if (!dfn[to])

{

tarjan(to,cur);

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

if (low[to]>dfn[cur]) isbridge[v[cur][i]]=true;

}

else if (to!=father) low[cur]=min(low[cur],low[to]);

}

}

void dfs(int cur,int par)

{

visited[cur]=true;

anc[cur]=par;

int i,to;

for (i=0;i<v[cur].size();i++)

{

if (isbridge[v[cur][i]]) continue;

to=(edge[v[cur][i]].x==cur?edge[v[cur][i]].y:edge[v[cur][i]].x);

if (!visited[to]) dfs(to,par);

}

}

// Tarjan求点双连通分量

// 注意该模板由圆方树改编来，只将大小大于1的双连通分量计入vector

const int MAXN=1e5;

struct Edge

{

int u,v,c;

}edge[MAXN+48];

vector<int> v[MAXN+48],vv[MAXN+48];

int dfn[MAXN+48],low[MAXN+48],Index=0,ind=n;

stack<int> s;

inline bool check(int id,Pair E)

{

if (edge[id].x==E.x && edge[id].y==E.y) return true;

if (edge[id].x==E.y && edge[id].y==E.x) return true;

return false;

}

inline void tarjan(int cur,int father)

{

dfn[cur]=low[cur]=++Index;int i,j,x,y,id,u;

for (i=0;i<int(v[cur].size());i++)

{

y=(edge[v[cur][i]].x==cur?edge[v[cur][i]].y:edge[v[cur][i]].x);

if (!dfn[y])

{

s.push(v[cur][i]);tarjan(y,cur);

low[cur]=min(low[cur],low[y]);

if (low[y]>=dfn[cur])

{

if (check(s.top(),mp(cur,y))) continue;

++ind;int pre=cur;

do

{

id=s.top();s.pop();

x=edge[id].x;u=edge[id].y;

if (x==pre) swap(x,u);

vv[ind].pb(x);

pre=x;

}

while (!check(id,mp(cur,y)));

}

}

else if (y!=father && dfn[y]<dfn[cur])

{

s.push(v[cur][i]);

low[cur]=min(low[cur],low[y]);

}

}

}