食物链(NOI2001)

/\* 当前节点与父亲结点的关系： 0 代表与父亲结点同类 1 代表被父亲结点吃 2 代表吃父亲结点 我们开一个数组re[i]，记录i号动物的种类情况。 种类有三种，分别记为0，1，2，1吃0，2吃1，0吃2。 对于每一个并查集，我们取根节点的类型为0. \*/

#include<cstdio>

#include<iostream>

#define M 50010

using namespace std;

int re[M],fa[M],n,m;

int find(int x){

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

int t=fa[x];

fa[x]=find(fa[x]);

re[x]=(re[x]+re[t])%3; //利用x的直接父亲t来更新x

return fa[x];

}

/\*建立关系时，先将两个子树的树根建立关系，之后两个数根与其子结点的关系 在find函数中更新\*/

void build(int x,int y,int r){

int a=find(x),b=find(y);

fa[a]=b; re[a]=(re[y]-re[x]+r+3)%3;

}

int main(){

freopen("jh.in","r",stdin);

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

int ans=0;

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

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

int f,x,y;

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

if(x>n||y>n){

ans++;

continue;

}

if(f==1)//同类

{

if(find(x)!=find(y)) build(x,y,0);

else if(re[x]!=re[y])ans++;

} else//x?y,??re[x]?re[y]?1

{ //x吃y，所以re[x]比re[y]大1

if(find(x)!=find(y)) build(x,y,1);

else if(re[x]!=(re[y]+1)%3)ans++;

}

}

printf("%d",ans);

return 0;

}

用father[n],path[n]数组分别记录当前结点的祖先和到祖先的距离。

这里规定距离为0时为同类，为1时表示被祖先吃，为2时表示吃祖先。

初始时每个元素的祖先是自己，距离为0。

对于每一句话，首先判断是x,y是否大于n，是否出现自己吃自己的情况，满足时讨论如下两种情况。

如果d=1，表示读入的x和y是同类，这时分别找到x,y和祖先fx,fy，如果fx=fy，说明他们是同一祖先。这时判断x和y到祖先的距离是否相等，显然，不相等证明这是一句假话。如果fx<>fy，说明x和y不在同一集合中，此时将这两个集合合并。合并时可以通过一个简单的向量关系算出fx->fy的距离，即path[fx]=path[y]-path[x].

如果d=2，表示x吃y，同样的找到它们的祖先，若fx=fy，则根据向量关系判断它们的距离是否矛盾，即检查path[x]-path[y]-2是否为0。若fx<>fy，则类似地根据已有的向量关系算出fx->fy的距离，即 path[fx]=path[y]-path[x]+2.

注意的地方是，因为path在运算过程可能出现负数，为避免这一情况且保证path的性质，可以在每次对path运算后加三再对三取模。

\*path数组在维护时要注意，建立新关系时，直接把fx指向fy，相应地修改path值即可，对儿子结点的维护放在getfather函数中进行。

var

fa,d:array[1..500000]of longint;

n,k,dd,x,y,i,ans,p,q:longint;

function find(x:longint):longint;

begin

if fa[x]=0 then exit(x);

find:=find(fa[x]);

d[x]:=(d[x]+d[fa[x]])mod 3;

fa[x]:=find;

end;

begin

readln(n,k);

fillchar(fa,sizeof(fa),0);

ans:=0;

for i:=1 to k do begin

readln(dd,x,y);

if (x>n) or (y>n) then begin

inc(ans);

continue;

end;

p:=find(x);

q:=find(y);

if dd=1 then begin

if p=q then begin

if d[x]<>d[y] then begin

inc(ans);

continue;

end;

end

else begin

fa[p]:=q;

d[p]:=(d[y]-d[x]+3) mod 3;

end;

end

else begin

if p=q then begin

if (2-d[x]+d[y]) mod 3<>0 then begin

inc(ans);

continue;

end;

end

else begin

fa[p]:=q;

d[p]:=(d[y]-d[x]+2) mod 3;

end;

end;

end;

writeln(ans);

end.