并查集：

题意 : 有两个不同的帮派,每个帮派至少有一个人. 判断两个人是否属于同一个帮派.

给你 N 个人,编号从 1 到 N,操作 M 次.

每次操作输入一个字符和两个数 x ,y

如果字符为 A 则判断 x 和 y 是否属于同一个帮派,并且输出结果.

如果字符为 D 则明确告诉你 x 和 y 是属于不同帮派的.

#include**<cstdio>**

**const int** N = 100000+10;

**int** p[N]; *//*存父亲节点

**int** r[N]; *//*存与根节点的关系，*0* 代表同类， *1*代表不同类

**int** find(**int** x) *//*找根节点

{

**if**(x == p[x]) **return** x;

**int** t = p[x]; *//*记录父亲节点方便下面更新*r[]*

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

r[x] = (r[x]+r[t])%2; *//*根据子节点与父亲节点的关系和父节点与爷爷节点的关系，推导子节点与爷爷节点的关系

**return** p[x]; *//*容易忘记

}

**void** Union(**int** x, **int** y)

{

**int** fx = find(x),fy = find(y);

p[fx] = fy; *//*合并

r[fx] = (r[x]+1+r[y])%2; *//fx*与*x*关系 *+ x*与*y*的关系 *+ y*与*fy*的关系 *= fx*与*fy*的关系

}

**void** set(**int** n)

{

**for**(**int** x = 1; x <= n; x++)

{

p[x] = x; *//*自己是自己的父节点

r[x] = 0; *//*自己和自己属于同一类

}

}

**int** main()

{

**int** T;

**int** n, m;

scanf(**"%d"**, &T);

**while**(T--)

{

scanf(**"%d%d%\*c"**, &n, &m);

set(n);

**char** c;

**int** x, y;

**while**(m--)

{

scanf(**"%c%d%d%\*c"**, &c, &x, &y); *//*注意输入

**if**(c == **'A'**)

{

**if**(find(x) == find(y)) *//*如果根节点相同，则表示能判断关系

{

**if**(r[x] != r[y]) printf(**"In different gangs.\n"**);

**else** printf(**"In the same gang.\n"**);

}

**else** printf(**"Not sure yet.\n"**);

}

**else if**(c == **'D'**)

{

Union(x, y);

}

}

}

**return** 0;

}

POJ 1182：

#include<cstdio>

#include<algorithm>

#include<cstring>

#include<cmath>

#include<queue>

#define inf 2e8+500

typedef long long ll;

using namespace std;

const int N=5e4+5;

int p[N],r[N];

void init(int n)

{

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

p[i]=i;

r[i]=0;

}

}

int find(int x)

{

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

int t=p[x];

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

r[x]=(r[t]+r[x])%3;

return p[x];

}

void unit(int x,int y,int d)

{

int fx=find(x);

int fy=find(y);

if(fx==fy) return ;

p[fy]=fx;

r[fy]=(3-r[y]+d-1+r[x])%3;//两点到根距离的差值加上两点距离的差值为两根之间的距离

}

int main()

{

int n,k,x,y,d;

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

init(n);

int ans=0;

while(k--){

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

if(x>n||y>n||(x==y&&d==2)) ans++;

else {

if(find(x)==find(y)){

if(d==1&&r[x]!=r[y]) ans++;

else if(d==2&&(r[x]+1)%3!=r[y]) ans++;

}

else unit(x,y,d);

}

}

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

return 0;

}