D. Cutting a graph

2 seconds

256 megabytes

standard input

standard output

There is an undirected graph and a sequence of operations of two types in the following format:

* cut *u* *v* — remove edge *u* - *v* from the graph;
* ask *u* *v* — check whether vertices *u* and *v* are in the same connected component.

After all the operations are applied, the graph contains no edges. Please, find the result of each operation of type ask.

**Input**

First line of input consists of three integers *n*, *m* and *k* (1 ≤ *n* ≤ 50 000, 0 ≤ *m* ≤ 100 000, *m* ≤ *k* ≤ 150 000) — the number of vertices in the graph, the number of edges and the number of operations, respectively .

Each of next *m* lines consists of two integers *ui* and *vi* (1 ≤ *ui*,  *vi* ≤ *n*) — ends of edge *i*. Vertices are numbered from 1, graph has no loops and multiple edges.

Each of the next *k* lines describes an operation in the following format:

* "cut *u* *v*" (1 ≤ *u*,  *v* ≤ *n*) — remove an edge between vertices *u* and *v*
* "ask *u* *v*" (1 ≤ *u*,  *v* ≤ *n*) — check whether vertices *u* and *v* are in the same component

Each edge is mentioned in operations of type cut once.

**Output**

For each of operation of type ask output "YES", if two given vertices are in the same component, and "NO" — otherwise. The order of the answers should correspond to the order of operations of type ask in input.

**Example**

**input**

**Copy**

3 3 7  
1 2  
2 3  
3 1  
ask 3 3  
cut 1 2  
ask 1 2  
cut 1 3  
ask 2 1  
cut 2 3  
ask 3 1

**output**

**Copy**

YES  
YES  
NO  
NO

#include<bits/stdc++.h>

#define int long long int

#define pb push\_back

#define pp pop\_back

#define pii pair<int,int>

#define vec vector<int>

#define mp make\_pair // DISJOINT SET UNION(DSU)

#define inf 1e10

#define F first

#define S second

using namespace std;

int parent[300001],rak[300001];

void make\_set(int v) // making a new set

{

parent[v]=v;

rak[v]=0;

}

int find\_set(int v) // finding the leader of the set containing v

{

if(parent[v]==v)

return v;

else

return parent[v]=find\_set(parent[v]);

}

void union\_set(int u,int v) // DOING UNION BY RANK

{

u=find\_set(u); // finding leader of the set containing u

v=find\_set(v); // finding leader of the set containing v

if(u!=v)

{

if(rak[u]<rak[v])

swap(u,v);

parent[v]=u;

if(rak[u]==rak[v])

rak[u]++;

}

}

int32\_t main()

{

ios\_base::sync\_with\_stdio(false);

cin.tie(NULL);

cout.tie(NULL);

int tt=1;

//cin>>tt;

while(tt--)

{

int n,m,k;

cin>>n>>m>>k;

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

make\_set(i);

vector<pii>edge;

map<pii,int> h;

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

{

int u,v;

cin>>u>>v;

if(u>v)

swap(u,v);

edge.pb({u,v});

h[{u,v}]=1;

}

vector<pair<string,pair<int,int>>> ve;

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

{

string s;

int a,b;

cin>>s>>a>>b;

if(a>b)

swap(a,b);

ve.pb({s,{a,b}});

if(s=="cut")

h[{a,b}]=0;

}

for(int i=0;i<(int)edge.size();i++)

{

if(h[edge[i]])

union\_set(edge[i].F,edge[i].S);

}

vector<string> ans;

for(int i=((int)ve.size())-1;i>=0;i--)

{

if(ve[i].F=="ask")

{

int a=find\_set(ve[i].S.F);

int b=find\_set(ve[i].S.S);

if(a==b)

ans.pb("YES");

else

ans.pb("NO");

}

else

{

union\_set(ve[i].S.F,ve[i].S.S);

}

}

for(int i=((int)ans.size())-1;i>=0;i--)

cout<<ans[i]<<"\n";

}

}