# Poufen

3237

#include<cstdio>

#include<cstring>

#include<iostream>

#include<algorithm>

#include<vector>

using namespace std;

const int M = 12345;

struct Edge{

int to,val,key;//边的另一个结点，权值，编号

Edge(int to=0,int val=0,int key=0):to(to),val(val),key(key){}

};

int fa[M][15];

vector<Edge>G[M];

int s[M];//子树结点的个数

int top[M];//重链的头结点

int e[M];//边在线段树中对应的标号

int pos[M];//点在线段树中对应的标号

int deep[M];//结点深度

struct Tree{

int vala,valb;//vala是最大值,valb是最小值

int tag;

}tree[M\*4];

int n;

int build(int num,int l,int r){

tree[num].vala=-M\*M;

tree[num].valb=M\*M;

tree[num].tag=0;

if(l==r) return 0;

int mid=(l+r)>>1;

build(num<<1,l,mid);

build(num<<1|1,mid+1,r);

return 0;

}

int pushup(int num){

tree[num].vala=max(tree[num<<1].vala,tree[num<<1|1].vala);

tree[num].valb=min(tree[num<<1].valb,tree[num<<1|1].valb);

return 0;

}

int change(int &x,int &y){

int tmp=x;x=-y;y=-tmp;

return 0;

}

int pushdown(int num,int l,int r){

if(l==r||tree[num].tag==0) return 0;

tree[num].tag=0;

int lson=num<<1,rson=num<<1|1;

tree[lson].tag^=1;tree[rson].tag^=1;

change(tree[lson].vala,tree[lson].valb);

change(tree[rson].vala,tree[rson].valb);

return 0;

}

int query(int num,int l,int r,int left,int right){

pushdown(num,l,r);

if(l>=left&&r<=right){

return tree[num].vala;

}

int mid=(l+r)>>1;

int ans=-M\*M;

if(left<=mid) ans=max(ans,query(num<<1,l,mid,left,right));

if(right>mid) ans=max(ans,query(num<<1|1,mid+1,r,left,right));

return ans;

}

int modefy(int num,int l,int r,int K,int val){//单点修改K值

if(l==r){

tree[num].vala=val;

tree[num].valb=val;

return 0;

}

pushdown(num,l,r);

int mid=(l+r)>>1;

if(K<=mid) modefy(num<<1,l,mid,K,val);

else modefy(num<<1|1,mid+1,r,K,val);

pushup(num);

return 0;

}

int rever(int num,int l,int r,int left,int right){//left 到 right区间取反

if(l>=left&&r<=right){

tree[num].tag^=1;

change(tree[num].vala,tree[num].valb);

return 0;

}

pushdown(num,l,r);

int mid=(l+r)>>1;

if(left<=mid) rever(num<<1,l,mid,left,right);

if(right>mid) rever(num<<1|1,mid+1,r,left,right);

pushup(num);

return 0;

}

int dfs1(int num){

s[num]=1;

for(int i=0;i<G[num].size();i++){

int to=G[num][i].to;

if(to!=fa[num][0]){

fa[to][0]=num;

deep[to]=deep[num]+1;

s[num]+=dfs1(to);

}

}

return s[num];

}

int sz=0;

int dfs2(int num,int chain){

top[num]=chain;

pos[num]=++sz;

int K=0;//找重儿子

for(int i=0;i<G[num].size();i++){

int to=G[num][i].to;

if(to!=fa[num][0]){

if(s[to]>s[K]) K=to;

}

else {

e[G[num][i].key]=pos[num];

modefy(1,1,n,pos[num],G[num][i].val);

//由于传统的树链剖分都是点权，这里是边权，

//所以将边权赋给deep较大的点，查询的时候需要注意

}

}

if(!K) return 0;

dfs2(K,chain);//优先递归重儿子，使得重链

for(int i=0;i<G[num].size();i++){

int to=G[num][i].to;

if(to!=fa[num][0]&&to!=K){

dfs2(to,to);

}

}

return 0;

}

int solve1(int num,int father){

int mx=-M\*M;

while(top[num]!=top[father]){

mx=max(mx,query(1,1,n,pos[top[num]],pos[num]));

num=fa[top[num]][0];

}

if(pos[father]+1<=pos[num])

mx=max(mx,query(1,1,n,pos[father]+1,pos[num]));

//由于传统的树链剖分都是点权，这里是边权，

//对于一条边的两个点，要查询deep较大的点，所以 pos[father]+1

return mx;

}

int solve2(int num,int father){

while(top[num]!=top[father]){

rever(1,1,n,pos[top[num]],pos[num]);

num=fa[top[num]][0];

}

if(pos[father]+1<=pos[num])

rever(1,1,n,pos[father]+1,pos[num]);

return 0;

}

int bin[15];//保存倍增值

int lca(int x,int y){

if(deep[x]<deep[y])swap(x,y);

int t=deep[x]-deep[y];

for(int i=0;i<=13;i++)

if(t&bin[i])x=fa[x][i];

for(int i=13;i>=0;i--)

if(fa[x][i]!=fa[y][i])

x=fa[x][i],y=fa[y][i];

if(x==y)return x;

return fa[x][0];

}

int addedge(int u,int v,int w,int k){

G[u].push\_back(Edge(v,w,k));

G[v].push\_back(Edge(u,w,k));

return 0;

}

inline int read(){

int tmp=0,flag=1;char ch=getchar();

while(ch>'9'||ch<'0'){if(ch=='-')flag=-1;ch=getchar();}

while(ch>='0'&&ch<='9'){tmp=tmp\*10+ch-'0';ch=getchar();}

return tmp\*flag;

}

int main(){

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

bin[0]=1;

for(int i=1;i<15;i++)bin[i]=bin[i-1]<<1;

int T;

cin>>T;

while(T--){

sz=0;

for(int i=1;i<=n;i++) G[i].clear();

memset(fa,0,sizeof(fa));

n=read();

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

int u=read(),v=read(),w=read();

addedge(u,v,w,i);

}

fa[1][0]=0;

build(1,1,n);

dfs1(1);

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

for(int j=1;j<=13;j++)

fa[i][j]=fa[fa[i][j-1]][j-1];

dfs2(1,1);

char ch[10];

while(scanf("%s",ch+1)){

if(ch[1]=='D')break;

int x=read(),y=read();

if(ch[1]=='Q'){

int father=lca(x,y);

printf("%d\n",max(solve1(x,father),solve1(y,father)));

}

if(ch[1]=='C'){

modefy(1,1,n,e[x],y);

}

if(ch[1]=='N'){

int father=lca(x,y);

solve2(x,father);solve2(y,father);

}

}

}

return 0;

}

6031

#include<bits/stdc++.h>

using namespace std;

const int M = 100010;

struct Edge{

int numn,next;

}e[M\*2];

int head[M];

int deep[M];

int fa[M][20];

int s[M];//子树结点的个数

int top[M];//重链的头结点

int pos[M];//点在线段树中对应的标号

struct Tree{

int val;

int tag;

int ans;

}tree[M\*4];

int n;

inline int build(int num,int l,int r){

tree[num].val=0;

tree[num].tag=0;

tree[num].ans=0;

if(l==r) return 0;

int mid=(l+r)>>1;

build(num<<1,l,mid);

build(num<<1|1,mid+1,r);

return 0;

}

inline int pushup(int num){

tree[num].val=max(tree[num<<1].val,tree[num<<1|1].val);

tree[num].ans=max(tree[num<<1].ans,tree[num<<1|1].ans);

return 0;

}

inline int insert(int num,int l,int r,int K,int val){

if(l==r){

tree[num].val=val;

tree[num].tag=0;

tree[num].ans=0;

return 0;

}

int mid=(l+r)>>1;

if(K<=mid) insert(num<<1,l,mid,K,val);

else insert(num<<1|1,mid+1,r,K,val);

pushup(num);

return 0;

}

inline int change(int num){

if(tree[num].tag==1) tree[num].ans=tree[num].val;

if(tree[num].tag==2) tree[num].ans=0;

return 0;

}

inline int pushdown(int num,int l,int r){

if(l==r||tree[num].tag==0) return 0;

int lson=num<<1,rson=num<<1|1;

tree[lson].tag=tree[num].tag;

tree[rson].tag=tree[num].tag;

tree[num].tag=0;

change(lson);change(rson);

return 0;

}

int u[M];

inline int query(int num,int l,int r,int left,int right){

if(tree[num].tag==1){

return u[min(r,right)];

}

if(tree[num].tag==2){

return 0;

}

pushdown(num,l,r);

if(l>=left&&r<=right){

return tree[num].ans;

}

int mid=(l+r)>>1;

int ans=-M;

if(left<=mid) ans=max(ans,query(num<<1,l,mid,left,right));

if(right>mid) ans=max(ans,query(num<<1|1,mid+1,r,left,right));

return ans;

}

inline int modefy(int num,int l,int r,int left,int right,int ta){//left 到 right区间取反

if(l>=left&&r<=right){

tree[num].tag=ta;

change(num);

return 0;

}

pushdown(num,l,r);

int mid=(l+r)>>1;

if(left<=mid) modefy(num<<1,l,mid,left,right,ta);

if(right>mid) modefy(num<<1|1,mid+1,r,left,right,ta);

pushup(num);

return 0;

}

int dfs1(int num){

s[num]=1;

for(int i=head[num];i!=-1;i=e[i].next){

int to=e[i].numn;

if(to!=fa[num][0]){

fa[to][0]=num;

deep[to]=deep[num]+1;

s[num]+=dfs1(to);

}

}

return s[num];

}

int sz=0;

int dfs2(int num,int chain,int deep){

top[num]=chain;

pos[num]=++sz;

insert(1,1,n,pos[num],deep);

u[pos[num]]=deep;

int K=0;

for(int i=head[num];i!=-1;i=e[i].next){

int to=e[i].numn;

if(to!=fa[num][0]){

if(s[to]>s[K]) K=to;

}

}

if(!K) return 0;

dfs2(K,chain,deep+1);

for(int i=head[num];i!=-1;i=e[i].next){

int to=e[i].numn;

if(to!=fa[num][0]&&to!=K){

dfs2(to,to,deep+1);

}

}

return 0;

}

inline int solve1(int num,int father){

int mx=-M;

while(top[num]!=top[father]){

mx=max(mx,query(1,1,n,pos[top[num]],pos[num]));

num=fa[top[num]][0];

}

if(pos[father]<=pos[num])

mx=max(mx,query(1,1,n,pos[father],pos[num]));

return mx;

}

inline int solve2(int num,int father,int tag){

while(top[num]!=top[father]){

modefy(1,1,n,pos[top[num]],pos[num],tag);

num=fa[top[num]][0];

}

if(pos[father]<=pos[num])

modefy(1,1,n,pos[father],pos[num],tag);

return 0;

}

int bin[20];

int lca(int x,int y){

if(deep[x]<deep[y])swap(x,y);

int t=deep[x]-deep[y];

for(int i=0;i<=18;i++)

if(t&bin[i])x=fa[x][i];

for(int i=18;i>=0;i--)

if(fa[x][i]!=fa[y][i])

x=fa[x][i],y=fa[y][i];

if(x==y)return x;

return fa[x][0];

}

int esz;

inline int read(){

int tmp=0,flag=1;char ch=getchar();

while(ch>'9'||ch<'0'){if(ch=='-')flag=-1;ch=getchar();}

while(ch>='0'&&ch<='9'){tmp=tmp\*10+ch-'0';ch=getchar();}

return tmp\*flag;

}

int t[M];

int main(){

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

bin[0]=1;

for(int i=1;i<19;i++)bin[i]=bin[i-1]<<1;

int m;

while(cin>>n>>m){

sz=0;

esz=0;

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

s[i]=0;

head[i]=-1;

}

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

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

int u=read(),v=read();

++esz;

e[esz].next=head[u];

head[u]=esz;

e[esz].numn=v;

++esz;

e[esz].next=head[v];

head[v]=esz;

e[esz].numn=u;

}

fa[1][0]=0;

build(1,1,n);

dfs1(1);

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

for(int j=1;j<=18;j++)

fa[i][j]=fa[fa[i][j-1]][j-1];

dfs2(1,1,1);

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

cin>>t[0];

for(int j=1;j<=t[0];j++){

t[j]=read();

solve2(t[j],1,1);

}

int x,y;

x=read();;

int Ans=0;

for(int j=1;j<=x;j++){

y=read();

Ans=max(Ans,solve1(y,1));

}

for(int j=1;j<=t[0];j++){

solve2(t[j],1,2);

}

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

}

}

return 0;

}