**CONVEX HULL TRICK**

struct Line{

ll m,h;

Line(){}

Line(ll \_m,ll \_h){

m = \_m;

h = \_h;

}

};

struct CHT { // for minimum (for maximum just change the sign of lines)

vector<Line> c;

int pos=0;

ll in(Line a, Line b){

ll x=b.h-a.h,y=a.m-b.m;

return x/y+(x%y?!((x>0)^(y>0)):0); // ==ceil(x/y)

}

void add(ll m, ll h){ // m's should be non increasing

Line l=(Line){m,h};

if(c.size()&&m==c.back().m){

l.h=min(h,c.back().h);c.pop\_back();if(pos)pos--;

}

while(c.size()>1&&in(c.back(),l)<=in(c[c.size()-2],c.back())){

c.pop\_back();if(pos)pos--;

}

c.pb(l);

}

inline bool fbin(ll x, int m){return in(c[m],c[m+1])>x;}

ll eval(ll x){

// O(log n) query:

int s=0,e=c.size();

while(e-s>1){

int m=(s+e)/2;

if(fbin(x,m-1))e=m;

else s=m;

}

return c[s].m\*x+c[s].h;

// O(1) query (for ordered x's):

while(pos>0&&fbin(x,pos-1))pos--;

while(pos<c.size()-1&&!fbin(x,pos))pos++;

return c[pos].m\*x+c[pos].h;

}

} CONVEX;

**TEXT 1**

// Codeforces 319C - AC

// http://codeforces.com/problemset/problem/319/C

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define fore(i,a,b) for(int i=a,ThxDem=b;i<ThxDem;++i)

using namespace std;

typedef long long ll;

typedef ll tc;

struct Line{tc m,h;};

struct CHT { // for minimum (for maximum just change the sign of lines)

vector<Line> c;

int pos=0;

tc in(Line a, Line b){

tc x=b.h-a.h,y=a.m-b.m;

return x/y+(x%y?!((x>0)^(y>0)):0); // ==ceil(x/y)

}

void add(tc m, tc h){ // m's should be non increasing

Line l=(Line){m,h};

if(c.size()&&m==c.back().m){

l.h=min(h,c.back().h);c.pop\_back();if(pos)pos--;

}

while(c.size()>1&&in(c.back(),l)<=in(c[c.size()-2],c.back())){

c.pop\_back();if(pos)pos--;

}

c.pb(l);

}

inline bool fbin(tc x, int m){return in(c[m],c[m+1])>x;}

tc eval(tc x){

// O(1) query (for ordered x's):

while(pos>0&&fbin(x,pos-1))pos--;

while(pos<c.size()-1&&!fbin(x,pos))pos++;

return c[pos].m\*x+c[pos].h;

}

};

ll a[100005];

ll b[100005];

int n;

ll f;

int main(){

scanf("%d",&n);

fore(i,0,n){int t;scanf("%d",&t);a[i]=t;}

fore(i,0,n){int t;scanf("%d",&t);b[i]=t;}

CHT ch;

ch.add(b[0],0);

fore(i,1,n){

f=ch.eval(a[i]);

ch.add(b[i],f);

}

printf("%lld\n",f);

return 0;

}

**TEST 2**

#include<bits/stdc++.h>

#define pb(x) push\_back(x)

using namespace std;

typedef long long ll;

const int N = (3e5);

const ll MOD = (1e9+7);

struct Line{

ll m,h;

Line(){}

Line(ll \_m,ll \_h){

m = \_m;

h = \_h;

}

};

struct CHT { // for minimum (for maximum just change the sign of lines)

vector<Line> c;

int pos=0;

ll in(Line a, Line b){

ll x=b.h-a.h,y=a.m-b.m;

return x/y+(x%y?!((x>0)^(y>0)):0); // ==ceil(x/y)

}

void add(ll m, ll h){ // m's should be non increasing

Line l=(Line){m,h};

if(c.size()&&m==c.back().m){

l.h=min(h,c.back().h);c.pop\_back();if(pos)pos--;

}

while(c.size()>1&&in(c.back(),l)<=in(c[c.size()-2],c.back())){

c.pop\_back();if(pos)pos--;

}

c.pb(l);

}

inline bool fbin(ll x, int m){return in(c[m],c[m+1])>x;}

ll eval(ll x){

// O(log n) query:

int s=0,e=c.size();

while(e-s>1){int m=(s+e)/2;

if(fbin(x,m-1))e=m;

else s=m;

}

return c[s].m\*x+c[s].h;

// O(1) query (for ordered x's):

while(pos>0&&fbin(x,pos-1))pos--;

while(pos<c.size()-1&&!fbin(x,pos))pos++;

return c[pos].m\*x+c[pos].h;

}

} CONVEX;

struct data{

ll q,a,b,r,d;

data(){}

data(ll \_q,ll \_a,ll \_b,ll \_r,ll \_d){

q = \_q;

a = \_q;

b = \_b;

r = \_r;

d = \_d;

}

void read(){

cin>>q>>a>>b>>r>>d;

}

};

data IN[N+2];

int main(){

int n;cin>>n;

for(int i=1;i<=n;i++) IN[i].read();

ll ans = 0;

for(int i=n;i>0;i--){

CONVEX.add(i,IN[i].r);

ll m = IN[i].q + i\*IN[i].d - CONVEX.eval(IN[i].d);

if(m<0) continue;

ll k = max(0LL,(m-IN[i].a)/IN[i].b);

m %= MOD;

k %= MOD;

ll cua = k\*(k+1)/2;

cua %= MOD;

ans += ( ((m-IN[i].a)\*k)%MOD - (IN[i].b\*cua)%MOD );

ans %= MOD;

ans += MOD;

ans %= MOD;

}

cout<<ans<<'\n';

}

**LINK CUT TREE**

typedef struct item \*pitem;

struct item {

int pr;bool rev;

pitem l,r,f,d;

item():pr(rand()),l(0),r(0),f(0),d(0),rev(0){}

};

void push(pitem t){

if(t&&t->rev){

swap(t->l,t->r);

if(t->l)t->l->rev^=1;

if(t->r)t->r->rev^=1;

t->rev=0;

}

}

void merge(pitem& t, pitem l, pitem r){

push(l);push(r);

if(!l||!r)t=l?l:r;

else if(l->pr>r->pr)merge(l->r,l->r,r),l->r->f=t=l;

else merge(r->l,l,r->l),r->l->f=t=r;

}

void push\_all(pitem t){

if(t->f)push\_all(t->f);

push(t);

}

void split(pitem t, pitem& l, pitem& r){

push\_all(t);

l=t->l;r=t->r;t->l=t->r=0;

while(t->f){

pitem f=t->f;t->f=0;

if(t==f->l){

if(r)r->f=f;

f->l=r;r=f;

}

else {

if(l)l->f=f;

f->r=l;l=f;

}

t=f;

}

if(l)l->f=0;

if(r)r->f=0;

}

pitem path(pitem p){return p->f?path(p->f):p;}

pitem tail(pitem p){push(p);return p->r?tail(p->r):p;}

pitem expose(pitem p){

pitem q,r,t;

split(p,q,r);

if(q)tail(q)->d=p;

merge(p,p,r);

while(t=tail(p),t->d){

pitem d=t->d;t->d=0;

split(d,q,r);

if(q)tail(q)->d=d;

merge(p,p,d);merge(p,p,r);

}

return p;

}

pitem root(pitem v){return tail(expose(v));}

void evert(pitem v){expose(v)->rev^=1;v->d=0;}

void link(pitem v, pitem w){ // make v son of w

evert(v);

pitem p=path(v);

merge(p,p,expose(w));

}

void cut(pitem v){ // cut v from its father

pitem p,q;

expose(v);split(v,p,q);v->d=0;

}

void cut(pitem v, pitem w){evert(w);cut(v);}

**TEST-LINK CUT TREE**

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define fore(i,a,b) for(int i=a,to=b;i<to;++i)

using namespace std;

typedef long long ll;

const int N=(1e5);

typedef struct item \*pitem;

struct item {

int pr;bool rev;

pitem l,r,f,d;

item():pr(rand()),l(0),r(0),f(0),d(0),rev(0){}

};

void push(pitem t){

if(t&&t->rev){

swap(t->l,t->r);

if(t->l)t->l->rev^=1;

if(t->r)t->r->rev^=1;

t->rev=0;

}

}

void merge(pitem& t, pitem l, pitem r){

push(l);push(r);

if(!l||!r)t=l?l:r;

else if(l->pr>r->pr)merge(l->r,l->r,r),l->r->f=t=l;

else merge(r->l,l,r->l),r->l->f=t=r;

}

void push\_all(pitem t){

if(t->f)push\_all(t->f);

push(t);

}

void split(pitem t, pitem& l, pitem& r){

push\_all(t);

l=t->l;r=t->r;t->l=t->r=0;

while(t->f){

pitem f=t->f;t->f=0;

if(t==f->l){

if(r)r->f=f;

f->l=r;r=f;

}

else {

if(l)l->f=f;

f->r=l;l=f;

}

t=f;

}

if(l)l->f=0;

if(r)r->f=0;

}

pitem path(pitem p){return p->f?path(p->f):p;}

pitem tail(pitem p){push(p);return p->r?tail(p->r):p;}

pitem expose(pitem p){

pitem q,r,t;

split(p,q,r);

if(q)tail(q)->d=p;

merge(p,p,r);

while(t=tail(p),t->d){

pitem d=t->d;t->d=0;

split(d,q,r);

if(q)tail(q)->d=d;

merge(p,p,d);merge(p,p,r);

}

return p;

}

pitem root(pitem v){return tail(expose(v));}

void evert(pitem v){expose(v)->rev^=1;v->d=0;}

void link(pitem v, pitem w){

evert(v);

pitem p=path(v);

merge(p,p,expose(w));

}

void cut(pitem v){

pitem p,q;

expose(v);split(v,p,q);v->d=0;

}

void cut(pitem v, pitem w){evert(w);cut(v);}

pitem x[100005];

int n,m;

int main(){

int n;cin>>n;

for(int i=0;i<n;i++) x[i]=new item();

string query;

getline(cin,query);

while(1){

getline(cin,query);

if(query=="E") break;

stringstream ss(query);

char type;int a,b;

ss>>type>>a>>b;

a--;b--;

if(type=='C') link(x[a],x[b]);

else if(type=='D') cut(x[a],x[b]);

else cout<<(root(x[a])==root(x[b])?"YES":"NO")<<endl;

}

return 0;

}

**WAVELET TREE**

struct WT {

vector<int> wt[1<<20];int n;

void init(int k, int s, int e){

if(s+1==e)return;

wt[k].clear();wt[k].pb(0);

int m=(s+e)/2;

init(2\*k,s,m);init(2\*k+1,m,e);

}

void add(int k, int s, int e, int v){

if(s+1==e)return;

int m=(s+e)/2;

if(v<m)wt[k].pb(wt[k].back()),add(2\*k,s,m,v);

else wt[k].pb(wt[k].back()+1),add(2\*k+1,m,e,v);

}

int query0(int k, int s, int e, int a, int b, int i){

if(s+1==e)return s;

int m=(s+e)/2;

int q=(b-a)-(wt[k][b]-wt[k][a]);

if(i<q)return query0(2\*k,s,m,a-wt[k][a],b-wt[k][b],i);

else return query0(2\*k+1,m,e,wt[k][a],wt[k][b],i-q);

}

void upd(int k, int s, int e, int i){

if(s+1==e)return;

int m=(s+e)/2;

int v0=wt[k][i+1]-wt[k][i],v1=wt[k][i+2]-wt[k][i+1];

if(!v0&&!v1)upd(2\*k,s,m,i-wt[k][i]);

else if(v0&&v1)upd(2\*k+1,m,e,wt[k][i]);

else if(v0)wt[k][i+1]--;

else wt[k][i+1]++;

}

void init(int \_n){n=\_n;init(1,0,n);} // (values in range [0,n))

void add(int v){add(1,0,n,v);}

int query0(int a, int b, int i){ // ith element in range [a,b)

return query0(1,0,n,a,b,i); // (if it was sorted)

}

void upd(int i){ // swap positions i,i+1

upd(1,0,n,i);

}

};

**WAVELET TREE TEST**

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define fore(i,a,b) for(int i=a,to=b;i<to;++i)

using namespace std;

typedef long long ll;

struct WT {

vector<int> wt[1<<20];int n;

void init(int k, int s, int e){

if(s+1==e)return;

wt[k].clear();wt[k].pb(0);

int m=(s+e)/2;

init(2\*k,s,m);init(2\*k+1,m,e);

}

void add(int k, int s, int e, int v){

if(s+1==e)return;

int m=(s+e)/2;

if(v<m)wt[k].pb(wt[k].back()),add(2\*k,s,m,v);

else wt[k].pb(wt[k].back()+1),add(2\*k+1,m,e,v);

}

int query0(int k, int s, int e, int a, int b, int i){

if(s+1==e)return s;

int m=(s+e)/2;

int q=(b-a)-(wt[k][b]-wt[k][a]);

if(i<q)return query0(2\*k,s,m,a-wt[k][a],b-wt[k][b],i);

else return query0(2\*k+1,m,e,wt[k][a],wt[k][b],i-q);

}

void upd(int k, int s, int e, int i){

if(s+1==e)return;

int m=(s+e)/2;

int v0=wt[k][i+1]-wt[k][i],v1=wt[k][i+2]-wt[k][i+1];

if(!v0&&!v1)upd(2\*k,s,m,i-wt[k][i]);

else if(v0&&v1)upd(2\*k+1,m,e,wt[k][i]);

else if(v0)wt[k][i+1]--;

else wt[k][i+1]++;

}

void init(int \_n){n=\_n;init(1,0,n);} // (values in range [0,n))

void add(int v){add(1,0,n,v);}

int query0(int a, int b, int i){ // ith element in range [a,b)

return query0(1,0,n,a,b,i); // (if it was sorted)

}

void upd(int i){ // swap positions i,i+1

upd(1,0,n,i);

}

} wt;

vector<int> z[1<<20];

int n,q,c,k,x[1<<20];

pair<int,int> xx[1<<20];

int main(){

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

fore(i,0,n){

scanf("%d",&k);

xx[i]=mp(k,i);

}

sort(xx,xx+n);

c=0;

fore(i,0,n){

if(i>0&&xx[i].fst!=xx[i-1].fst)c++;

x[xx[i].snd]=c;

}

c++;

wt.init(c);

fore(i,0,n)wt.add(x[i]),z[x[i]].pb(i);

while(q--){

int t;

scanf("%d",&t);

int i,l,k;

if(t==0){

scanf("%d%d%d",&i,&l,&k);i++;l--;k--;

int d=wt.query0(0,i,k);

if(l>=z[d].size())puts("-1");

else printf("%d\n",z[d][l]);

}

else {

scanf("%d",&i);

if(x[i]==x[i+1])continue;

int k=lower\_bound(z[x[i]].begin(),z[x[i]].end(),i)-z[x[i]].begin();

z[x[i]][k]++;

k=lower\_bound(z[x[i+1]].begin(),z[x[i+1]].end(),i+1)-z[x[i+1]].begin();

z[x[i+1]][k]--;

wt.upd(i);swap(x[i],x[i+1]);

}

}

return 0;

}

**DINIC – FLOW**

// Min cut: nodes with dist>=0 vs nodes with dist<0

// Matching MVC: left nodes with dist<0 + right nodes with dist>0

int nodes,src,dst; // remember to init nodes

int dist[MAXN],q[MAXN],work[MAXN];

struct edge {int to,rev;ll f,cap;};

vector<edge> g[MAXN];

void add\_edge(int s, int t, ll cap){

g[s].pb((edge){t,SZ(g[t]),0,cap});

g[t].pb((edge){s,SZ(g[s])-1,0,0});

}

bool dinic\_bfs(){

fill(dist,dist+nodes,-1);dist[src]=0;

int qt=0;q[qt++]=src;

for(int qh=0;qh<qt;qh++){

int u=q[qh];

fore(i,0,SZ(g[u])){

edge &e=g[u][i];int v=g[u][i].to;

if(dist[v]<0&&e.f<e.cap)dist[v]=dist[u]+1,q[qt++]=v;

}

}

return dist[dst]>=0;

}

ll dinic\_dfs(int u, ll f){

if(u==dst)return f;

for(int &i=work[u];i<SZ(g[u]);i++){

edge &e=g[u][i];

if(e.cap<=e.f)continue;

int v=e.to;

if(dist[v]==dist[u]+1){

ll df=dinic\_dfs(v,min(f,e.cap-e.f));

if(df>0){e.f+=df;g[v][e.rev].f-=df;return df;}

}

}

return 0;

}

ll max\_flow(int \_src, int \_dst){

src=\_src;dst=\_dst;

ll result=0;

while(dinic\_bfs()){

fill(work, work+nodes, 0);

while(ll delta=dinic\_dfs(src,INF))result+=delta;

}

return result;

}

**DINIC –TEST 1**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

#define pb push\_back

#define fore(i,a,b) for(int i=a,to=b;i<to;i++)

#define SZ(x) (int) x.size()

const int N = (1000);

const int MAXN = (2\*N+5);

const ll INF = (1e12);

// Min cut: nodes with dist>=0 vs nodes with dist<0

// Matching MVC: left nodes with dist<0 + right nodes with dist>0

struct edge {int to,rev;ll f,cap;};

struct Dinic{

int nodes,src,dst; // remember to init nodes

int dist[MAXN],q[MAXN],work[MAXN];

vector<edge> g[MAXN];

Dinic(){}

Dinic(int \_nodes,int \_src,int \_dst){

nodes = \_nodes;src = \_src;dst = \_dst;

}

void add\_edge(int s, int t, ll cap){

g[s].pb((edge){t,SZ(g[t]),0,cap});

g[t].pb((edge){s,SZ(g[s])-1,0,0});

}

bool dinic\_bfs(){

fill(dist,dist+nodes,-1);dist[src]=0;

int qt=0;q[qt++]=src;

for(int qh=0;qh<qt;qh++){

int u=q[qh];

fore(i,0,SZ(g[u])){

edge &e=g[u][i];int v=g[u][i].to;

if(dist[v]<0&&e.f<e.cap)dist[v]=dist[u]+1,q[qt++]=v;

}

}

return dist[dst]>=0;

}

ll dinic\_dfs(int u, ll f){

if(u==dst)return f;

for(int &i=work[u];i<SZ(g[u]);i++){

edge &e=g[u][i];

if(e.cap<=e.f)continue;

int v=e.to;

if(dist[v]==dist[u]+1){

ll df=dinic\_dfs(v,min(f,e.cap-e.f));

if(df>0){e.f+=df;g[v][e.rev].f-=df;return df;}

}

}

return 0;

}

ll max\_flow(){

ll result=0;

while(dinic\_bfs()){

fill(work, work+nodes, 0);

while(ll delta=dinic\_dfs(src,INF))result+=delta;

}

return result;

}

};

int n,m,t,tot;

int A[N+2],D[N+2];

vector< pair<int,int> > g[N+2];

bool can(int time){

Dinic dinic(n+m+2,0,n+m+1);

for(int i=1;i<=n;i++) dinic.add\_edge(0,i,A[i]);

for(int i=1;i<=m;i++) dinic.add\_edge(i+n,n+m+1,D[i]);

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

for(int j=0;j<SZ(g[i]);j++){

if(g[i][j].second>time) continue;

dinic.add\_edge(i,g[i][j].first+n,INF);

}

}

int res = dinic.max\_flow();

if(res==tot) return true;

return false;

}

int main(){

ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

cin>>n>>m>>t;

for(int i=1;i<=n;i++) cin>>A[i];

tot = accumulate(A+1,A+n+1,0);

for(int i=1;i<=m;i++) cin>>D[i];

int a,b,c;

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

cin>>a>>b>>c;

g[a].pb(make\_pair(b,c));

}

int lo=1,hi=(1e6);

if(!can(hi)){

cout<<"-1\n";

return 0;

}

while((hi-lo)>1){

int mi = (hi+lo)/2;

if(can(mi)) hi=mi;

else lo=mi;

}

cout<<hi<<'\n';

return 0;

}

**DINIC – TEST 2**

**RECONSTRUCYENDO EL FLUJO**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

#define fore(i,a,b) for(int i=a,to=b;i<to;++i)

#define pb push\_back

#define SZ(x) (int)x.size()

// Min cut: nodes with dist>=0 vs nodes with dist<0

// Matching MVC: left nodes with dist<0 + right nodes with dist>0

const int N = (100);

int nodes,src,dst; // remember to init nodes

int dist[2\*N+5],q[2\*N+5],work[2\*N+5];

const ll INF = (1e12);

struct edge {int to,rev;ll f,cap;};

vector<edge> g[2\*N+5];

void add\_edge(int s, int t, ll cap){

g[s].pb((edge){t,SZ(g[t]),0,cap});

g[t].pb((edge){s,SZ(g[s])-1,0,0});

}

bool dinic\_bfs(){

fill(dist,dist+nodes,-1);dist[src]=0;

int qt=0;q[qt++]=src;

for(int qh=0;qh<qt;qh++){

int u=q[qh];

fore(i,0,SZ(g[u])){

edge &e=g[u][i];int v=g[u][i].to;

if(dist[v]<0&&e.f<e.cap)dist[v]=dist[u]+1,q[qt++]=v;

}

}

return dist[dst]>=0;

}

ll dinic\_dfs(int u, ll f){

if(u==dst)return f;

for(int &i=work[u];i<SZ(g[u]);i++){

edge &e=g[u][i];

if(e.cap<=e.f)continue;

int v=e.to;

if(dist[v]==dist[u]+1){

ll df=dinic\_dfs(v,min(f,e.cap-e.f));

if(df>0){e.f+=df;g[v][e.rev].f-=df;return df;}

}

}

return 0;

}

ll max\_flow(int \_src, int \_dst){

src=\_src;dst=\_dst;

ll result=0;

while(dinic\_bfs()){

fill(work, work+nodes, 0);

while(ll delta=dinic\_dfs(src,INF))result+=delta;

}

return result;

}

int n,m;

ll A[N+2],B[N+2];

ll M[N+2][N+2];

int main(){

cin>>n>>m;

for(int i=1;i<=n;i++) cin>>A[i];

for(int i=1;i<=n;i++) cin>>B[i];

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

add\_edge(0,i,A[i]);

add\_edge(i+n,2\*n+1,B[i]);

add\_edge(i,i+n,INF);

}

while(m--){

int a,b;

cin>>a>>b;

add\_edge(a,b+n,INF);

add\_edge(b,a+n,INF);

}

nodes = 2\*n + 2;

src = 0;

dst = 2\*n+1;

ll val = max\_flow(src,dst);

if(val == accumulate(A+1,A+n+1,0LL) && val == accumulate(B+1,B+n+1,0LL)){

cout<<"YES\n";

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

for(int j=0;j<SZ(g[i]);j++){

int fin = g[i][j].to-n;

if(fin<=0) continue;

M[i][fin] += g[i][j].f;

}

}

for(int i=1;i<=n;i++) for(int j=1;j<=n;j++) cout<<M[i][j]<<(char)(j==n?10:32);

}else cout<<"NO\n";

return 0;

}

**MAX FLOW MIN COST**

typedef ll tf;const tf INFFLUJO=1e14;

typedef ll tc;const tc INFCOSTO=1e14;

struct edge {

int u,v;tf cap,flow;tc cost;

tf rem(){return cap-flow;}

};

int nodes; // remember to init nodes

vector<int> g[MAXN];

vector<edge> e;

void add\_edge(int u, int v, tf cap, tc cost) {

g[u].pb(SZ(e));e.pb((edge){u,v,cap,0,cost});

g[v].pb(SZ(e));e.pb((edge){v,u,0,0,-cost});

}

tc dist[MAXN],mncost;

int pre[MAXN];

tf cap[MAXN],mxflow;

bool in\_queue[MAXN];

void flow(int s, int t){

memset(in\_queue,0,sizeof(in\_queue));

mxflow=mncost=0;

while(1){

fill(dist,dist+nodes,INFCOSTO);dist[s]=0;

memset(pre,-1,sizeof(pre));pre[s]=0;

memset(cap,0,sizeof(cap));cap[s]=INFFLUJO;

queue<int> q;q.push(s);in\_queue[s]=1;

while(SZ(q)){

int u=q.front();q.pop();in\_queue[u]=0;

fore(\_,0,SZ(g[u])){

int i=g[u][\_];

edge &E=e[i];

if(E.rem()&&dist[E.v]>dist[u]+E.cost+1e-9){

dist[E.v]=dist[u]+E.cost;

pre[E.v]=i;

cap[E.v]=min(cap[u],E.rem());

if(!in\_queue[E.v])q.push(E.v),in\_queue[E.v]=1;

}

}

}

if(pre[t]<0)break;

mxflow+=cap[t];mncost+=cap[t]\*dist[t];

for(int v=t;v!=s;v=e[pre[v]].u){

e[pre[v]].flow+=cap[t];e[pre[v]^1].flow-=cap[t];

}

}

}

**MAX FLOW MIN COST – TEST**

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define SZ(x) int((x).size())

#define fore(i,a,b) for(int i=a,to=b;i<to;++i)

using namespace std;

typedef long long ll;

#define MAXN 512

typedef int tf;const tf INFFLUJO=1e9;

typedef int tc;const tc INFCOSTO=1e9;

struct edge {

int u,v;tf cap,flow;tc cost;

tf rem(){return cap-flow;}

};

int nodes; // remember to init nodes

vector<int> g[MAXN];vector<edge> e;

void add\_edge(int u, int v, tf cap, tc cost) {

g[u].pb(SZ(e));e.pb((edge){u,v,cap,0,cost});

g[v].pb(SZ(e));e.pb((edge){v,u,0,0,-cost});

}

tc dist[MAXN],mncost;int pre[MAXN];tf cap[MAXN],mxflow;bool in\_queue[MAXN];

void flow(int s, int t){

memset(in\_queue,0,sizeof(in\_queue));

mxflow=mncost=0;

while(1){

fill(dist,dist+nodes,INFCOSTO);dist[s]=0;

memset(pre,-1,sizeof(pre));pre[s]=0;

memset(cap,0,sizeof(cap));cap[s]=INFFLUJO;

queue<int> q;q.push(s);in\_queue[s]=1;

while(SZ(q)){

int u=q.front();q.pop();in\_queue[u]=0;

fore(\_,0,SZ(g[u])){

int i=g[u][\_];edge &E=e[i];

if(E.rem()&&dist[E.v]>dist[u]+E.cost+1e-9){

dist[E.v]=dist[u]+E.cost;pre[E.v]=i;

cap[E.v]=min(cap[u],E.rem());

if(!in\_queue[E.v])q.push(E.v),in\_queue[E.v]=1;

}

}

}

if(pre[t]<0)break;

mxflow+=cap[t];mncost+=cap[t]\*dist[t];

for(int v=t;v!=s;v=e[pre[v]].u){

e[pre[v]].flow+=cap[t];e[pre[v]^1].flow-=cap[t];

}

}

}

int q[512];

int n;

int main(){

int tn;

scanf("%d",&tn);

while(tn--){

scanf("%d",&n);

nodes=2+n;

memset(q,0,sizeof(q));

fore(i,0,n){

int x;

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

q[x]++;

}

fore(i,0,n)if(q[i]>0)add\_edge(0,2+i,q[i],0);

fore(i,0,n)add\_edge(2+i,1,1,0);

int m;

scanf("%d",&m);

while(m--){

int x,y;

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

add\_edge(2+x,2+y,512,1);

add\_edge(2+y,2+x,512,1);

}

flow(0,1);

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

fore(i,0,nodes)g[i].clear();

e.clear();

}

return 0;

}

**GEOMETRY**

**PUNTO**

struct pt { // for 3D add z coordinate

double x,y;

pt(double x, double y):x(x),y(y){}

pt(){}

double norm2(){return \*this\*\*this;}

double norm(){return sqrt(norm2());}

bool operator==(pt p){return abs(x-p.x)<EPS&&abs(y-p.y)<EPS;}

pt operator+(pt p){return pt(x+p.x,y+p.y);}

pt operator-(pt p){return pt(x-p.x,y-p.y);}

pt operator\*(double t){return pt(x\*t,y\*t);}

pt operator/(double t){return pt(x/t,y/t);}

double operator\*(pt p){return x\*p.x+y\*p.y;}

// pt operator^(pt p){ // only for 3D

// return pt(y\*p.z-z\*p.y,z\*p.x-x\*p.z,x\*p.y-y\*p.x);}

double angle(pt p){ // redefine acos for values out of range

return acos(\*this\*p/(norm()\*p.norm()));}

pt unit(){return \*this/norm();}

double operator%(pt p){return x\*p.y-y\*p.x;}

// 2D from now on

bool operator<(pt p)const{ // for convex hull

return x<p.x-EPS||(abs(x-p.x)<EPS&&y<p.y-EPS);}

bool left(pt p, pt q){ // is it to the left of directed line pq?

return (q-p)%(\*this-p)>EPS;}

pt rot(pt r){return pt(\*this%r,\*this\*r);}

pt rot(double a){return rot(pt(sin(a),cos(a)));}

};

pt ccw90(1,0);

pt cw90(-1,0);

**LINE**

int sgn2(double x){return x<0?-1:1;}

struct ln {

pt p,pq;

ln(pt p, pt q):p(p),pq(q-p){}

ln(){}

bool has(pt r){return dist(r)<EPS;}

bool seghas(pt r){return has(r)&&(r-p)\*(r-(p+pq))-EPS<0;}

// bool operator /(ln l){return (pq.unit()^l.pq.unit()).norm()<EPS;} // 3D

bool operator/(ln l){return abs(pq.unit()%l.pq.unit())<EPS;} // 2D

bool operator==(ln l){return \*this/l&&has(l.p);}

pt operator^(ln l){ // intersection

if(\*this/l)return pt(DINF,DINF);

//FOR DOUBLES

pt r=l.p+l.pq\*((p-l.p)%pq/(l.pq%pq));

// if(!has(r)){return pt(NAN,NAN,NAN);} // check only for 3D

return r;

//FOR INTEGER

ll a=(p-l.p)%pq;

ll b=l.pq%pq;ll bb=b;

ll sx=l.pq.x,sy=l.pq.y;

ll g=gcd(sx,b);

sx/=g;b/=g;

if(a%b)return pt(DINF,DINF);

sx\*=a/b;

b=bb;

g=gcd(sy,b);

sy/=g;b/=g;

if(a%b)return pt(DINF,DINF);

sy\*=a/b;

pt r=l.p+pt(sx,sy);

return r;

}

double angle(ln l){return pq.angle(l.pq);}

int side(pt r){return has(r)?0:sgn2(pq%(r-p));} // 2D

pt proj(pt r){return p+pq\*((r-p)\*pq/pq.norm2());}

pt ref(pt r){return proj(r)\*2-r;}

double dist(pt r){return (r-proj(r)).norm();}

// double dist(ln l){ // only 3D

// if(\*this/l)return dist(l.p);

// return abs((l.p-p)\*(pq^l.pq))/(pq^l.pq).norm();

// }

ln rot(auto a){return ln(p,p+pq.rot(a));} // 2D

};

ln bisector(ln l, ln m){ // angle bisector

pt p=l^m;

return ln(p,p+l.pq.unit()+m.pq.unit());

}

ln bisector(pt p, pt q){ // segment bisector (2D)

return ln((p+q)\*.5,p).rot(ccw90);

}

**PLANE**

struct plane {

pt a,n; // n: normal unit vector

plane(pt a, pt b, pt c):a(a),n(((b-a)^(c-a)).unit()){}

plane(){}

bool has(pt p){return abs((p-a)\*n)<EPS;}

double angle(plane w){return acos(n\*w.n);}

double dist(pt p){return abs((p-a)\*n);}

pt proj(pt p){inter(ln(p,p+n),p);return p;}

bool inter(ln l, pt& r){

double x=n\*(l.p+l.pq-a),y=n\*(l.p-a);

if(abs(x-y)<EPS)return false;

r=(l.p\*x-(l.p+l.pq)\*y)/(x-y);

return true;

}

bool inter(plane w, ln& r){

pt nn=n^w.n;pt v=n^nn;double d=w.n\*v;

if(abs(d)<EPS)return false;

pt p=a+v\*(w.n\*(w.a-a)/d);

r=ln(p,p+nn);

return true;

}

};

**POLYGON**

int sgn(double x){return x<-EPS?-1:x>EPS;}

struct pol {

int n;vector<pt> p;

pol(){}

pol(vector<pt> \_p){p=\_p;n=p.size();}

double area(){

double r=0.;

fore(i,0,n)r+=p[i]%p[(i+1)%n];

return abs(r)/2; // negative if CW, positive if CCW

}

pt centroid(){ // (barycenter)

pt r(0,0);double t=0;

fore(i,0,n){

r=r+(p[i]+p[(i+1)%n])\*(p[i]%p[(i+1)%n]);

t+=p[i]%p[(i+1)%n];

}

return r/t/3;

}

bool has(pt q){ // O(n)

fore(i,0,n)if(ln(p[i],p[(i+1)%n]).seghas(q))return true;

int cnt=0;

fore(i,0,n){

int j=(i+1)%n;

int k=sgn((q-p[j])%(p[i]-p[j]));

int u=sgn(p[i].y-q.y),v=sgn(p[j].y-q.y);

if(k>0&&u<0&&v>=0)cnt++;

if(k<0&&v<0&&u>=0)cnt--;

}

return cnt!=0;

}

void normalize(){ // (call before haslog, remove collinear first)

if(p[2].left(p[0],p[1]))reverse(p.begin(),p.end());

int pi=min\_element(p.begin(),p.end())-p.begin();

vector<pt> s(n);

fore(i,0,n)s[i]=p[(pi+i)%n];

p.swap(s);

}

bool haslog(pt q){ // O(log(n)) only CONVEX. Call normalize first

if(q.left(p[0],p[1])||q.left(p.back(),p[0]))return false;

int a=1,b=p.size()-1; // returns true if point on boundary

while(b-a>1){ // (change sign of EPS in left

int c=(a+b)/2; // to return false in such case)

if(!q.left(p[0],p[c]))a=c;

else b=c;

}

return !q.left(p[a],p[a+1]);

}

pt farthest(pt v){ // O(log(n)) only CONVEX

if(n<10){

int k=0;

fore(i,1,n)if(v\*(p[i]-p[k])>EPS)k=i;

return p[k];

}

if(n==SZ(p))p.pb(p[0]);

pt a=p[1]-p[0];

int s=0,e=n,ua=v\*a>EPS;

if(!ua&&v\*(p[n-1]-p[0])<=EPS)return p[0];

while(1){

int m=(s+e)/2;pt c=p[m+1]-p[m];

int uc=v\*c>EPS;

if(!uc&&v\*(p[m-1]-p[m])<=EPS)return p[m];

if(ua&&(!uc||v\*(p[s]-p[m])>EPS))e=m;

else if(ua||uc||v\*(p[s]-p[m])>=-EPS)s=m,a=c,ua=uc;

else e=m;

assert(e>s+1);

}

}

pol cut(ln l){ // cut CONVEX polygon by line l

vector<pt> q; // returns part at left of l.pq

fore(i,0,n){

int d0=sgn(l.pq%(p[i]-l.p)),d1=sgn(l.pq%(p[(i+1)%n]-l.p));

if(d0>=0)q.pb(p[i]);

ln m(p[i],p[(i+1)%n]);

if(d0\*d1<0&&!(l/m))q.pb(l^m);

}

return pol(q);

}

double intercircle(circle c){ // area of intersection with circle

double r=0.;

fore(i,0,n){

int j=(i+1)%n;double w=c.intertriangle(p[i],p[j]);

if((p[j]-c.o)%(p[i]-c.o)>0)r+=w;

else r-=w;

}

return abs(r);

}

double callipers(){ // square distance of most distant points

double r=0; // prereq: convex, ccw, NO COLLINEAR POINTS

for(int i=0,j=n<2?0:1;i<j;++i){

for(;;j=(j+1)%n){

r=max(r,(p[i]-p[j]).norm2());

if((p[(i+1)%n]-p[i])%(p[(j+1)%n]-p[j])<=EPS)break;

}

}

return r;

}

};

// Dynamic convex hull trick

vector<pol> w;

void add(pt q){ // add(q), O(log^2(n))

vector<pt> p={q};

while(!w.empty()&&SZ(w.back().p)<2\*SZ(p)){

for(pt v:w.back().p)p.pb(v);

w.pop\_back();

}

w.pb(pol(chull(p)));

}

ll query(pt v){ // max(q\*v:q in w), O(log^2(n))

ll r=-INF;

for(auto& p:w)r=max(r,p.farthest(v)\*v);

return r;

}

**POLYGON TEST**

// Kattis pointinpolygon - AC

// https://open.kattis.com/problems/pointinpolygon

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define fore(i,a,b) for(int i=a,ThxDem=b;i<ThxDem;++i)

using namespace std;

typedef long long ll;

const double EPS=1e-7;

const double DINF=1e200;

struct pt { // for 3D add z coordinate

double x,y;

pt(double x, double y):x(x),y(y){}

pt(){}

double norm2(){return \*this\*\*this;}

double norm(){return sqrt(norm2());}

bool operator==(pt p){return abs(x-p.x)<EPS&&abs(y-p.y)<EPS;}

pt operator+(pt p){return pt(x+p.x,y+p.y);}

pt operator-(pt p){return pt(x-p.x,y-p.y);}

pt operator\*(double t){return pt(x\*t,y\*t);}

pt operator/(double t){return pt(x/t,y/t);}

double operator\*(pt p){return x\*p.x+y\*p.y;}

// pt operator^(pt p){ // only for 3D

// return pt(y\*p.z-z\*p.y,z\*p.x-x\*p.z,x\*p.y-y\*p.x);}

// double angle(pt p){ // redefine acos for values out of range

// return acos(\*this\*p/(norm()\*p.norm()));}

// pt unit(){return \*this/norm();}

double operator%(pt p){return x\*p.y-y\*p.x;}

// 2D from now on

// bool operator<(pt p)const{ // for convex hull

// return x<p.x-EPS||(abs(x-p.x)<EPS&&y<p.y-EPS);}

// bool left(pt p, pt q){ // is it to the left of directed line pq?

// return (q-p)%(\*this-p)>EPS;}

// pt rot(pt r){return pt(\*this%r,\*this\*r);}

// pt rot(double a){return rot(pt(sin(a),cos(a)));}

};

//pt ccw90(1,0);

//pt cw90(-1,0);

int sgn2(double x){return x<0?-1:1;}

struct ln {

pt p,pq;

ln(pt p, pt q):p(p),pq(q-p){}

ln(){}

bool has(pt r){return dist(r)<EPS;}

bool seghas(pt r){return has(r)&&(r-p)\*(r-(p+pq))-EPS<0;}

// bool operator /(ln l){return (pq.unit()^l.pq.unit()).norm()<EPS;} // 3D

// bool operator/(ln l){return abs(pq.unit()%l.pq.unit())<EPS;} // 2D

// bool operator==(ln l){return \*this/l&&has(l.p);}

// pt operator^(ln l){ // intersection

// if(\*this/l)return pt(DINF,DINF);

// pt r=l.p+l.pq\*((p-l.p)%pq/(l.pq%pq));

// if(!has(r)){return pt(NAN,NAN,NAN);} // check only for 3D

// return r;

// }

// double angle(ln l){return pq.angle(l.pq);}

// int side(pt r){return has(r)?0:sgn2(pq%(r-p));} // 2D

pt proj(pt r){return p+pq\*((r-p)\*pq/pq.norm2());}

// pt ref(pt r){return proj(r)\*2-r;}

double dist(pt r){return (r-proj(r)).norm();}

// double dist(ln l){ // only 3D

// if(\*this/l)return dist(l.p);

// return abs((l.p-p)\*(pq^l.pq))/(pq^l.pq).norm();

// }

// ln rot(auto a){return ln(p,p+pq.rot(a));} // 2D

};

//ln bisector(ln l, ln m){ // angle bisector

// pt p=l^m;

// return ln(p,p+l.pq.unit()+m.pq.unit());

//}

//ln bisector(pt p, pt q){ // segment bisector (2D)

// return ln((p+q)\*.5,p).rot(ccw90);

//}

int sgn(double x){return x<-EPS?-1:x>EPS;}

struct pol {

int n;vector<pt> p;

pol(){}

pol(vector<pt> \_p){p=\_p;n=p.size();}

int has(pt q){

fore(i,0,n)if(ln(p[i],p[(i+1)%n]).seghas(q))return 2; // minor change to distinguish on and in

int cnt=0;

fore(i,0,n){

int j=(i+1)%n;

int k=sgn((q-p[j])%(p[i]-p[j]));

int u=sgn(p[i].y-q.y),v=sgn(p[j].y-q.y);

if(k>0&&u<0&&v>=0)cnt++;

if(k<0&&v<0&&u>=0)cnt--;

}

return cnt!=0;

}

};

int main(){

int n;

while(scanf("%d",&n),n){

vector<pt> pp;

double x,y;

fore(i,0,n){

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

pp.pb(pt(x,y));

}

pol p(pp);

int m;

scanf("%d",&m);

while(m--){

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

int r=p.has(pt(x,y));

if(r==2)puts("on");

else if(r==1)puts("in");

else puts("out");

}

}

return 0;

}

**CIRCLE TEST**

// SPOJ TAP2015A - AC

// http://www.spoj.com/problems/TAP2015A/

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define fore(i,a,b) for(int i=a,ThxDem=b;i<ThxDem;++i)

using namespace std;

typedef long long ll;

const double DINF=numeric\_limits<double>::infinity();

const double EPS=1e-8;

struct pt { // for 3D add z coordinate

double x,y;

pt(double x, double y):x(x),y(y){}

pt(){}

double norm2(){return \*this\*\*this;}

double norm(){return sqrt(norm2());}

bool operator==(pt p){return abs(x-p.x)<EPS&&abs(y-p.y)<EPS;}

pt operator+(pt p){return pt(x+p.x,y+p.y);}

pt operator-(pt p){return pt(x-p.x,y-p.y);}

pt operator\*(double t){return pt(x\*t,y\*t);}

pt operator/(double t){return pt(x/t,y/t);}

double operator\*(pt p){return x\*p.x+y\*p.y;}

// pt operator^(pt p){ // only for 3D

// return pt(y\*p.z-z\*p.y,z\*p.x-x\*p.z,x\*p.y-y\*p.x);}

double angle(pt p){ // redefine acos for values out of range

return acos(\*this\*p/(norm()\*p.norm()));}

pt unit(){return \*this/norm();}

double operator%(pt p){return x\*p.y-y\*p.x;}

// 2D from now on

bool operator<(pt p)const{ // for convex hull

return x<p.x-EPS||(abs(x-p.x)<EPS&&y<p.y-EPS);}

bool left(pt p, pt q){ // is it to the left of directed line pq?

return (q-p)%(\*this-p)>EPS;}

pt rot(pt r){return pt(\*this%r,\*this\*r);}

pt rot(double a){return rot(pt(sin(a),cos(a)));}

};

pt ccw90(1,0);

pt cw90(-1,0);

struct circle {

pt o;double r;

circle(){}

circle(pt o, double r):o(o),r(r){}

// circle(pt x, pt y, pt z){o=bisector(x,y)^bisector(x,z);r=(o-x).norm();}

bool has(pt p){return (o-p).norm()<r+EPS;}

vector<pt> operator^(circle c){

vector<pt> s;

double d=(o-c.o).norm();

if(d>r+c.r+EPS||d+min(r,c.r)+EPS<max(r,c.r))return s;

double x=(d\*d-c.r\*c.r+r\*r)/(2\*d);

double y=sqrt(r\*r-x\*x);

pt v=(c.o-o)/d;

s.pb(o+v\*x+v.rot(ccw90)\*y);

if(y>EPS)s.pb(o+v\*x-v.rot(ccw90)\*y);

return s;

}

/\*

vector<pt> operator^(ln l){

vector<pt> s;

pt p=l.proj(o);

double d=(p-o).norm();

if(d-EPS>r)return s;

if(abs(d-r)<EPS){s.pb(p);return s;}

d=sqrt(r\*r-d\*d);

s.pb(p+l.pq.unit()\*d);

s.pb(p-l.pq.unit()\*d);

return s;

}

vector<pt> tang(pt p){

double d=sqrt((p-o).norm2()-r\*r);

return \*this^circle(p,d);

}

\*/

};

circle c[128];

int n;

int main(){

while(scanf("%d",&n)!=EOF){

fore(i,0,n){

int x,y,r;

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

c[i]=circle(pt(x,y),r);

}

int r=1;

fore(i,0,n){

fore(j,i+1,n){

auto v=c[i]^c[j];

for(auto p:v){

int s=0;

fore(k,0,n)s+=c[k].has(p);

r=max(r,s);

}

}

int s=0;

fore(k,0,n)s+=c[k].has(c[i].o);

r=max(r,s);

}

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

}

return 0;

}

**PUNTO ENTERES CUBIERTOS POR SEGMENTOS**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const ll INF = (1e9);

struct Segment{

ll x1, y1, x2, y2;

Segment(){}

ll manyPoint(){

ll dif1 = abs(x2-x1);

ll dif2 = abs(y2-y1);

return (\_\_gcd(dif1,dif2)+1);

}

};

bool in(int x, int l, int r){

if (l > r) swap(l, r);

return (l <= x && x <= r);

}

struct line{

ll A, B, C;

line(){};

line(Segment a){

A = a.y1 - a.y2;

B = a.x2 - a.x1;

C = -A \* a.x1 - B \* a.y1;

};

};

ll det(ll a, ll b, ll c, ll d){

return a \* d - b \* c;

}

bool inter(Segment a, Segment b, ll& x, ll& y){

line l1(a), l2(b);

ll dx = det(l1.C, l1.B, l2.C, l2.B);

ll dy = det(l1.A, l1.C, l2.A, l2.C);

ll d = det(l1.A, l1.B, l2.A, l2.B);

if (d == 0) return false;

if (dx % d != 0 || dy % d != 0) return false;

x = -dx / d;

y = -dy / d;

if (!in(x, a.x1, a.x2) || !in(y, a.y1, a.y2)) return false;

if (!in(x, b.x1, b.x2) || !in(y, b.y1, b.y2)) return false;

return true;

}

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

int n;cin>>n;

vector<Segment> v(n);

for(int i=0;i<n;i++){

cin>>v[i].x1>>v[i].y1>>v[i].x2>>v[i].y2;

}

ll ans = 0;

for(int i=0;i<n;i++){

ans += v[i].manyPoint();

set< pair<ll,ll> > repetidas;

ll x,y;

for(int j=0;j<i;j++){

if (inter(v[i], v[j], x, y)) repetidas.insert(make\_pair(x,y));

}

ans -= repetidas.size();

}

cout<<ans<<'\n';

return 0;

}

**PUNTO DENTRO DE UN POLIGONO ,ENTEROS**

#include<bits/stdc++.h>

#define Vector Point

using namespace std;

#define fore(i,a,b) for(int i=a,to=b;i<to;i++)

typedef long long ll;

struct Point{

ll x,y;

Point(){}

Point(ll \_x,ll \_y){

x = \_x;y = \_y;

}

ll mod2(){

return (x\*x+y\*y);

}

ll operator%(Point P){return x\*P.y-y\*P.x;}

};

Point operator +(const Point &a ,const Point &b){

return Point(a.x+b.x,a.y+b.y);

}

Point operator -(const Point &a ,const Point &b){

return Point(a.x-b.x,a.y-b.y);

}

bool operator <(const Point &a, const Point &b){

if(a.x != b.x) return a.x < b.x;

return a.y < b.y;

}

ll cross(const Vector &A, const Vector &B){

return A.x \* B.y - A.y \* B.x;

}

ll area(const Point &A, const Point &B, const Point &C) {

return cross(B - A, C - A);

}

vector <Point> ConvexHull(vector <Point> Poly){

sort(Poly.begin(),Poly.end());

int nP = Poly.size(),k = 0;

Point H[ 2\*nP ];

for( int i = 0 ; i < nP ; ++i ){

while( k >= 2 && area( H [ k - 2 ] , H[ k - 1 ] , Poly[ i] ) <= 0) --k;

H[ k++ ] = Poly[ i ];

}

for( int i = nP - 2 , t = k ; i >= 0 ; --i ){

while( k > t && area( H[ k - 2 ] , H[ k - 1 ] , Poly[ i ]) <= 0) --k;

H[ k++ ] = Poly[ i ];

}

if( k == 0 )return vector <Point>();

return vector <Point> ( H , H + k - 1 );

}

bool isInConvex(vector<Point>&P,Point &A){

int n = P.size(),lo=1,hi=P.size()-1;

if(area(P[0],P[1],A)<0) return 0;

if(area(P[n-1],P[0],A)<0) return 0;

while(hi-lo>1){

int mid = (hi+lo)/2;

if(area(P[0],P[mid],A) > 0) lo=mid;

else hi = mid;

}

return area(P[lo],P[hi],A)>=0;

}

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

int n;cin>>n;

vector<Point> v;

for(int i=0;i<n;i++){

Point cur;cin>>cur.x>>cur.y;

v.push\_back(cur);

}

vector<Point> w = ConvexHull(v);

int q;cin>>q;

int ans = 0;

while(q--){

Point query;cin>>query.x>>query.y;

if(isInConvex(w,query)){

ans++;

}

}

cout<<ans<<'\n';

return 0;

}

**ESTRUCTURAS SOBRE ARBOLES**

**HLD**

// SPOJ QTREE - AC

// http://www.spoj.com/problems/QTREE/

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define fore(i,a,b) for(int i=a,ThxDem=b;i<ThxDem;++i)

using namespace std;

typedef long long ll;

#define oper max

#define NEUT -(1<<30)

struct STree { // segment tree for min over integers

vector<int> st;int n;

STree(int n): st(4\*n+5,NEUT), n(n) {}

void init(int k, int s, int e, int \*a){

if(s+1==e){st[k]=a[s];return;}

int m=(s+e)/2;

init(2\*k,s,m,a);init(2\*k+1,m,e,a);

st[k]=oper(st[2\*k],st[2\*k+1]);

}

void upd(int k, int s, int e, int p, int v){

if(s+1==e){st[k]=v;return;}

int m=(s+e)/2;

if(p<m)upd(2\*k,s,m,p,v);

else upd(2\*k+1,m,e,p,v);

st[k]=oper(st[2\*k],st[2\*k+1]);

}

int query(int k, int s, int e, int a, int b){

if(s>=b||e<=a)return NEUT;

if(s>=a&&e<=b)return st[k];

int m=(s+e)/2;

return oper(query(2\*k,s,m,a,b),query(2\*k+1,m,e,a,b));

}

void init(int \*a){init(1,0,n,a);}

void upd(int p, int v){upd(1,0,n,p,v);}

int query(int a, int b){return query(1,0,n,a,b);}

}; // usage: STree rmq(n);rmq.init(x);rmq.upd(i,v);rmq.query(s,e);

#define MAXN 100005

vector<int> g[MAXN];

int wg[MAXN],dad[MAXN],dep[MAXN]; // weight,father,depth

void dfs1(int x){

wg[x]=1;

for(int y:g[x])if(y!=dad[x]){

dad[y]=x;dep[y]=dep[x]+1;dfs1(y);

wg[x]+=wg[y];

}

}

int curpos,pos[MAXN],head[MAXN];

void hld(int x, int c){

if(c<0)c=x;

pos[x]=curpos++;head[x]=c;

int mx=-1;

for(int y:g[x])if(y!=dad[x]&&(mx<0||wg[mx]<wg[y]))mx=y;

if(mx>=0)hld(mx,c);

for(int y:g[x])if(y!=mx&&y!=dad[x])hld(y,-1);

}

void hld\_init(){dad[0]=-1;dep[0]=0;dfs1(0);curpos=0;hld(0,-1);}

int query(int x, int y, STree& rmq){

int r=NEUT;

while(head[x]!=head[y]){

if(dep[head[x]]>dep[head[y]])swap(x,y);

r=oper(r,rmq.query(pos[head[y]],pos[y]+1));

y=dad[head[y]];

}

if(dep[x]>dep[y])swap(x,y); // now x is lca

r=oper(r,rmq.query(pos[x]+1,pos[y]+1)); // pos[x]+1 for not counting lca

return r;

}

// for updating: rmq.upd(pos[x],v);

int n;

int a[MAXN],b[MAXN],c[MAXN];

int z[MAXN];

int main(){

int tn;

scanf("%d",&tn);

while(tn--){

scanf("%d",&n);

STree rmq(n);

fore(i,0,n-1){

scanf("%d%d%d",a+i,b+i,c+i);a[i]--;b[i]--;

g[a[i]].pb(b[i]);g[b[i]].pb(a[i]);

}

hld\_init();

z[0]=NEUT;

fore(i,0,n-1){

int x=a[i],y=b[i];

if(x==dad[y])z[pos[y]]=c[i];

else z[pos[x]]=c[i];

}

rmq.init(z);

char t[16];

while(scanf("%s",t),t[0]!='D'){

int i,j;

scanf("%d%d",&i,&j);

if(t[0]=='C'){

int x=a[i-1],y=b[i-1];

if(x==dad[y])rmq.upd(pos[y],j);

else rmq.upd(pos[x],j);

}

else printf("%d\n",query(i-1,j-1,rmq));

}

fore(i,0,n)g[i].clear();

}

return 0;

}

// for updating: rmq.upd(pos[x],v);

**CENTROID DESCOMPOSITION**

// SPOJ QTREE5 - AC

// http://www.spoj.com/problems/QTREE5/

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define fore(i,a,b) for(int i=a,ThxDem=b;i<ThxDem;++i)

using namespace std;

typedef long long ll;

#define K 17

#define MAXN (1<<K)

vector<int> g[MAXN];int n;

bool tk[MAXN];

int fat[MAXN]; // father in centroid decomposition

int szt[MAXN]; // size of subtree

int calcsz(int x, int f){

szt[x]=1;

for(auto y:g[x])if(y!=f&&!tk[y])szt[x]+=calcsz(y,x);

return szt[x];

}

void cdfs(int x=0, int f=-1, int sz=-1){ // O(nlogn)

if(sz<0)sz=calcsz(x,-1);

for(auto y:g[x])if(!tk[y]&&szt[y]\*2>=sz){

szt[x]=0;cdfs(y,f,sz);return;

}

tk[x]=true;fat[x]=f;

for(auto y:g[x])if(!tk[y])cdfs(y,x);

}

void centroid(){memset(tk,false,sizeof(tk));cdfs();}

int F[K][1<<K],D[1<<K];

void lca\_dfs(int x){

fore(i,0,g[x].size()){

int y=g[x][i];if(y==F[0][x])continue;

F[0][y]=x;D[y]=D[x]+1;lca\_dfs(y);

}

}

void lca\_init(){

D[0]=0;F[0][0]=-1;

lca\_dfs(0);

fore(k,1,K)fore(x,0,n)

if(F[k-1][x]<0)F[k][x]=-1;

else F[k][x]=F[k-1][F[k-1][x]];

}

int lca(int x, int y){

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

for(int k=K-1;k>=0;--k)if(D[x]-(1<<k)>=D[y])x=F[k][x];

if(x==y)return x;

for(int k=K-1;k>=0;--k)if(F[k][x]!=F[k][y])x=F[k][x],y=F[k][y];

return F[0][x];

}

int dist(int x, int y){return D[x]+D[y]-2\*D[lca(x,y)];}

multiset<int> w[MAXN];

int c[MAXN];

int main(){

scanf("%d",&n);

fore(\_,1,n){

int x,y;

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

g[x].pb(y);g[y].pb(x);

}

lca\_init();

centroid();

int q;

scanf("%d",&q);

while(q--){

int t,x;

scanf("%d%d",&t,&x);x--;

if(!t){

c[x]^=1;

for(int y=x;y>=0;y=fat[y]){

if(c[x])w[y].insert(dist(x,y));

else w[y].erase(w[y].find(dist(x,y)));

}

}

else {

int r=1<<30;

for(int y=x;y>=0;y=fat[y]){

if(!w[y].empty())r=min(r,dist(x,y)+\*w[y].begin());

}

if(r==(1<<30))puts("-1");

else printf("%d\n",r);

}

}

return 0;

}

**LCA**

#include<bits/stdc++.h>

using namespace std;

#define fore(i,a,b) for(int i=a,to=b;i<to;i++)

#define pb push\_back

typedef long long ll;

const ll MOD = (998244353);

const int N = (4e4);

const int K = 18;

ll pot(ll x,ll y){

if(y==0) return 1LL;

if(y==1) return x;

ll ans = 1;

if(y&1) ans = x;

ll val = pot(x,y/2);

ans \*= val;

ans %= MOD;

ans \*= val;

ans %= MOD;

return ans;

}

ll inv(ll x){

return pot(x,MOD-2);

}

vector<int> g[1<<K];int n; // K such that 2^K>=n

vector<ll> c[1<<K];

int F[K][1<<K],D[1<<K],S[1<<K];

bool vis[1<<K];

void lca\_dfs(int x){

vis[x] = 1;

fore(i,0,g[x].size()){

int y=g[x][i];if(y==F[0][x])continue;

F[0][y]=x;D[y]=D[x]+1;S[y]=(S[x]\*c[x][i])%MOD;lca\_dfs(y);

}

}

void lca\_init(int x){

D[x]=0;F[0][x]=-1;S[x] = 1;

lca\_dfs(x);

fore(k,1,K)fore(x,0,n)

if(F[k-1][x]<0)F[k][x]=-1;

else F[k][x]=F[k-1][F[k-1][x]];

}

int lca(int x, int y){

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

for(int k=K-1;k>=0;--k)if(D[x]-(1<<k)>=D[y])x=F[k][x];

if(x==y)return x;

for(int k=K-1;k>=0;--k)if(F[k][x]!=F[k][y])x=F[k][x],y=F[k][y];

return F[0][x];

}

ll query(int x,int y){

int p = lca(x,y);

ll ans = S[x];

ans \*= S[y];

ans %= MOD;

ans \*= inv(S[p]);

ans %= MOD;

ans \*= inv(S[p]);

ans %= MOD;

return ans;

}

int pa[N+2];

int Find(int x){

return (pa[x]==x?x:pa[x]=Find(pa[x]));

}

void Union(int x,int y){

int xx = Find(x),yy = Find(y);

pa[xx] = yy;

}

bool same(int x,int y){

return Find(x)==Find(y);

}

int main(){

ios::sync\_with\_stdio(0);cin.tie(NULL);

int nn,q;cin>>nn;

map<string,int> M;

for(int i=0;i<N;i++) pa[i] = i;

int id = 0;

for(int i=0;i<nn;i++){

string a,b;cin>>a>>b;

if(!M.count(a))M[a]=id++;

if(!M.count(b))M[b]=id++;

ll x;cin>>x;

if(same(M[a],M[b])) continue;

Union(M[a],M[b]);

g[M[a]].pb(M[b]);

c[M[a]].pb(x);

g[M[b]].pb(M[a]);

c[M[b]].pb(inv(x));

}

n = id;

for(int i=0;i<n;i++)if(!vis[i])lca\_init(i);

cin>>q;

while(q--){

string a,b;

cin>>a>>b;

if(a==b) cout<<1<<'\n';

else if(!M.count(a)||!M.count(b)) cout<<"-1\n";

else if(!same(M[a],M[b])) cout<<"-1\n";

else{

cout<<query(M[a],M[b])<<'\n';

}

}

}

**LCA IMPERIAL ROAD LATIN AMERICA 2017**

#include<bits/stdc++.h>

using namespace std;

#define fore(i,a,b) for(int i=a,to=b;i<to;i++)

typedef long long ll;

const int N = (1e5), K=18;

vector<int> g[1<<K],cost[1<<K];int n; // K such that 2^K>=n

int F[K][1<<K],D[1<<K],C[K][1<<K],DD[K][1<<K];

int total;

void lca\_dfs(int x){

fore(i,0,g[x].size()){

int y=g[x][i];if(y==F[0][x])continue;

F[0][y]=x;D[y]=D[x]+1;lca\_dfs(y);

C[0][y]=cost[x][i];

DD[0][y] = cost[x][i];

}

}

void lca\_init(){

D[0]=0;F[0][0]=-1;

lca\_dfs(0);

fore(k,1,K)fore(x,0,n)

if(F[k-1][x]<0)F[k][x]=-1;

else F[k][x]=F[k-1][F[k-1][x]],C[k][x]=max(C[k-1][x],C[k-1][F[k-1][x]]),DD[k][x]=DD[k-1][x]+DD[k-1][F[k-1][x]];

}

int lca(int x, int y){

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

for(int k=K-1;k>=0;--k)if(D[x]-(1<<k)>=D[y])x=F[k][x];

if(x==y)return x;

for(int k=K-1;k>=0;--k)if(F[k][x]!=F[k][y])x=F[k][x],y=F[k][y];

return F[0][x];

}

int maxCost(int hijo,int padre){

if(hijo==padre) return 0;

int ans = 0;

for(int k=K-1;k>=0;--k){

if(D[hijo]-(1<<k)>=D[padre]){

ans = max(ans,C[k][hijo]);

hijo = F[k][hijo];

}

}

return ans;

}

map<pair<int,int>,int> pesos;

int query(int x,int y){

int padre = lca(x,y);

int maximo = max(maxCost(x,padre),maxCost(y,padre));

return total-maximo+pesos[make\_pair(x,y)];

}

struct edge{

int u,to;ll c;

edge(int \_u,int \_to,ll \_c){

u = \_u;to = \_to;c = \_c;

}

};

bool operator<(const edge &a,const edge &b){

return a.c<b.c;

}

int pa[N+2];

int Find(int x){

return (x==pa[x]?x:pa[x]=Find(pa[x]));

}

int Union(int x,int y){

int xx = Find(x),yy = Find(y);

pa[xx] = yy;

}

int main(){

ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

int r;cin>>n>>r;

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

int a,b;ll c;

vector<edge> prim;

while(r--){

cin>>a>>b>>c;

a--;b--;

prim.push\_back(edge(a,b,c));

pesos[make\_pair(a,b)]=c;

}

sort(prim.begin(),prim.end());

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

edge &cur = prim[i];

if(Find(cur.u)==Find(cur.to)) continue;

Union(cur.u,cur.to);

total += cur.c;

g[cur.u].push\_back(cur.to);

g[cur.to].push\_back(cur.u);

cost[cur.u].push\_back(cur.c);

cost[cur.to].push\_back(cur.c);

}

lca\_init();

int q;cin>>q;

while(q--){

cin>>a>>b;a--;b--;

cout<<query(a,b)<<'\n';

}

return 0;

}

**MATH**

**POLLARD RHO**

ll gcd(ll a, ll b){return a?gcd(b%a,a):b;}

ll mulmod(ll a, ll b, ll m) {

if(!b)return 0;

ll q=mulmod(a,b/2,m);q=(q+q)%m;

return b&1?(q+a)%m:q;

}

ll expmod(ll b, ll e, ll m){

if(!e)return 1;

ll q=expmod(b,e/2,m);q=mulmod(q,q,m);

return e&1?mulmod(b,q,m):q;

}

bool is\_prime\_prob(ll n, int a){

if(n==a)return true;

ll s=0,d=n-1;

while(d%2==0)s++,d/=2;

ll x=expmod(a,d,n);

if((x==1)||(x+1==n))return true;

fore(\_,0,s-1){

x=mulmod(x,x,n);

if(x==1)return false;

if(x+1==n)return true;

}

return false;

}

bool rabin(ll n){ // true iff n is prime

if(n==1)return false;

int ar[]={2,3,5,7,11,13,17,19,23};

fore(i,0,9)if(!is\_prime\_prob(n,ar[i]))return false;

return true;

}

ll rho(ll n){

if(!(n&1))return 2;

ll x=2,y=2,d=1;

ll c=rand()%n+1;

while(d==1){

x=(mulmod(x,x,n)+c)%n;

y=(mulmod(y,y,n)+c)%n;

y=(mulmod(y,y,n)+c)%n;

if(x>=y)d=gcd(x-y,n);

else d=gcd(y-x,n);

}

return d==n?rho(n):d;

}

void fact(ll n, map<ll,int>& f){ //O (lg n)^3

if(n==1)return;

if(rabin(n)){f[n]++;return;}

ll q=rho(n);fact(q,f);fact(n/q,f);

}

######PYTHON VERSION###

import sys

sys.setrecursionlimit(10000)

f = []

def expmod(b,e,m):

if(e==0):

return 1

q = expmod(b,e//2,m)

q = (q\*q)%m

if(e%2==1):

return (b\*q)%m

else:

return q

def is\_prime\_prob(n,a):

if(n==a):

return True

s=0

d=n-1

while(d%2==0):

s=s+1

d=d//2

x=expmod(a,d,n)

if(x==1 or x+1==n):

return True

for i in range(s-1):

x=(x\*x)%n

if(x==1):

return False

if(x+1==n):

return True

return False

def rabin(n):

if (n==1):

return False

ar = [2,3,5,7,11,13,17,19,23]

for i in range(len(ar)):

if(not is\_prime\_prob(n,ar[i])):

return False

return True

**STRINGS**

**KMP**

vector<int> kmppre(string& t){ // r[i]: longest border of t[0,i)

vector<int> r(t.size()+1);r[0]=-1;

int j=-1;

fore(i,0,t.size()){

while(j>=0&&t[i]!=t[j])j=r[j];

r[i+1]=++j;

}

return r;

}

void kmp(string& s, string& t){ // find t in s

int j=0;vector<int> b=kmppre(t);

fore(i,0,s.size()){

while(j>=0&&s[i]!=t[j])j=b[j];

if(++j==t.size())printf("Match at %d\n",i-j+1),j=b[j];

}

}

**MANACHER – MAXIMO PALINDROME CENTERED**

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define fore(i,a,b) for(int i=a,to=b;i<to;++i)

using namespace std;

typedef long long ll;

#define MAXN (1<<20)

int d1[MAXN];//d1[i]=max odd palin centered on i

int d2[MAXN];//d2[i]=max even palin centered on i

//s aabbaacaabbaa

//d1 1111117111111

//d2 20103010010301

void manacher(string& s){

int l=0,r=-1,n=s.size();

fore(i,0,n){

int k=i>r?1:min(d1[l+r-i],r-i);

while(i+k<n&&i-k>=0&&s[i+k]==s[i-k])k++;

d1[i]=k--;

if(i+k>r)l=i-k,r=i+k;

}

l=0;r=-1;

fore(i,0,n){

int k=i>r?0:min(d2[l+r-i+1],r-i+1);k++;

while(i+k<=n&&i-k>=0&&s[i+k-1]==s[i-k])k++;

d2[i]=--k;

if(i+k-1>r)l=i-k,r=i+k-1;

}

}

char \_s[MAXN];

int main(){

int k;

scanf("%d%s",&k,\_s);

string s(\_s);

manacher(s);

int r=0;

fore(i,0,s.size()){

if(k%2==1&&2\*d1[i]-1>=k)r++;

if(k%2==0&&2\*d2[i]>=k)r++;

}

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

return 0;

}

**SUFFIX AUTOMATON**

struct state {int len,link;map<char,int> next;}; //clear next!!

state st[100005];

int sz,last;

void sa\_init(){

last=st[0].len=0;sz=1;

st[0].link=-1;

}

void sa\_extend(char c){

int k=sz++,p;

st[k].len=st[last].len+1;

for(p=last;p!=-1&&!st[p].next.count(c);p=st[p].link)st[p].next[c]=k;

if(p==-1)st[k].link=0;

else {

int q=st[p].next[c];

if(st[p].len+1==st[q].len)st[k].link=q;

else {

int w=sz++;

st[w].len=st[p].len+1;

st[w].next=st[q].next;st[w].link=st[q].link;

for(;p!=-1&&st[p].next[c]==q;p=st[p].link)st[p].next[c]=w;

st[q].link=st[k].link=w;

}

}

last=k;

}

// input: abcbcbc

// i,link,len,next

// 0 -1 0 (a,1) (b,5) (c,7)

// 1 0 1 (b,2)

// 2 5 2 (c,3)

// 3 7 3 (b,4)

// 4 9 4 (c,6)

// 5 0 1 (c,7)

// 6 11 5 (b,8)

// 7 0 2 (b,9)

// 8 9 6 (c,10)

// 9 5 3 (c,11)

// 10 11 7

// 11 7 4 (b,8)

**TEST 1 K-TH SUBSTRING**

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define fore(i,a,b) for(int i=a,to=b;i<to;++i)

using namespace std;

typedef long long ll;

const int N = (1e5);

struct state {int len,link;map<char,int> next;}; //clear next!!

state st[2\*N+5];

ll memo[2\*N+5];

int sz,last;

void sa\_init(){

last=st[0].len=0;sz=1;

st[0].link=-1;

}

void sa\_extend(char c){

int k=sz++,p;

st[k].len=st[last].len+1;

for(p=last;p!=-1&&!st[p].next.count(c);p=st[p].link)st[p].next[c]=k;

if(p==-1)st[k].link=0;

else {

int q=st[p].next[c];

if(st[p].len+1==st[q].len)st[k].link=q;

else {

int w=sz++;

st[w].len=st[p].len+1;

st[w].next=st[q].next;st[w].link=st[q].link;

for(;p!=-1&&st[p].next[c]==q;p=st[p].link)st[p].next[c]=w;

st[q].link=st[k].link=w;

}

}

last=k;

}

ll dp(int x){

if(memo[x] != -1) return memo[x];

ll &ans = memo[x] = 1;

for(map<char,int>::iterator it=st[x].next.begin();it!=st[x].next.end();it++) ans += dp((\*it).second);

return ans;

}

string ans = "";

map<char,char> decode;

string alpha;

void kth(ll x,int pos,char y){

if(pos)ans.push\_back(alpha[y-'a']);

if(x==0) return;

state cur = st[pos];

ll act = 0,last = 0;

for(map<char,int>::iterator it=cur.next.begin();it!=cur.next.end();it++){

last = act;

act += memo[(\*it).second];

if(act>=x){

kth(x-last-1,(\*it).second,(\*it).first);

break;

}

}

}

vector<ll> kmppre(string& t){ // r[i]: longest border of t[0,i)

vector<ll> r(t.size()+1);r[0]=-1;

ll j=-1;

for(ll i=0;i<t.size();i++){

while(j>=0&&t[i]!=t[j])j=r[j];

r[i+1]=++j;

}

return r;

}

ll kmp(string& s, string& t){ // find t in s

ll j=0;vector<ll> b=kmppre(t);

ll ans = 0;

for(ll i=0;i<s.size();i++){

while(j>=0&&s[i]!=t[j])j=b[j];

if(++j==t.size())ans++,j=b[j];

}

return ans;

}

int main(){

string s;

while(cin>>s>>alpha){

memset(memo,-1,sizeof memo);

decode.clear();

for(int i=0;i<26;i++){

decode[alpha[i]] = char(i+'a');

}

sa\_init();

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

sa\_extend(decode[s[i]]);

}

ll tot = dp(0) - 1;

int q;cin>>q;

ll k;

while(q--){

cin>>k;

if(k>tot) cout<<"\*\n0\n";

else{

ans="";

kth(k,0,'\*');

cout<<ans<<'\n';

int numberOfOcurrences = kmp(s,ans);

cout<<numberOfOcurrences<<'\n';

}

}

for(int i=0;i<sz;i++) st[i].next.clear();

}

}

**TEST2 SUBSTRING**

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define fore(i,a,b) for(int i=a,to=b;i<to;++i)

using namespace std;

const int N = (1e5);

struct state {int len,link;map<char,int> next;}; //clear next!!

state st[N+5];

int sz,last;

void sa\_init(){

last=st[0].len=0;sz=1;

st[0].link=-1;

}

void sa\_extend(char c){

int k=sz++,p;

st[k].len=st[last].len+1;

for(p=last;p!=-1&&!st[p].next.count(c);p=st[p].link)st[p].next[c]=k;

if(p==-1)st[k].link=0;

else {

int q=st[p].next[c];

if(st[p].len+1==st[q].len)st[k].link=q;

else {

int w=sz++;

st[w].len=st[p].len+1;

st[w].next=st[q].next;st[w].link=st[q].link;

for(;p!=-1&&st[p].next[c]==q;p=st[p].link)st[p].next[c]=w;

st[q].link=st[k].link=w;

}

}

last=k;

}

bool substring(string s){

int i = 0, pos=0;

while(i<s.size()){

state cur = st[pos];

if(cur.next.find(s[i]) == cur.next.end()) return false;

pos = cur.next[s[i]];

i++;

}

return true;

}

int main(){

sa\_init();

string s;cin>>s;

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

sa\_extend(s[i]);

}

int q;cin>>q;

while(q--){

cin>>s;

if(substring(s)) cout<<"Y\n";

else cout<<"N\n";

}

}

**MAXIMO SUBSTRING COMUN A 2 STRING**

string lcs (string s, string t) {

sa\_init();

for (int i=0; i<(int)s.length(); ++i)

sa\_extend (s[i]);

int v = 0, l = 0,

best = 0, bestpos = 0;

for (int i=0; i<(int)t.length(); ++i) {

while (v && ! st[v].next.count(t[i])) {

v = st[v].link;

l = st[v].length;

}

if (st[v].next.count(t[i])) {

v = st[v].next[t[i]];

++l;

}

if (l > best)

best = l, bestpos = i;

}

return t.substr (bestpos-best+1, best);

}

**PROBLEMAS VARIOS**

**BIT + BINARY SEARCH**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int M = (1e6);

int N=1;

struct BIT{

ll tree[M+1];

BIT(){

for(int i=0;i<=M;i++) tree[i] = 0;

}

void Clear(){

for(int i=0;i<=4\*N;i++) tree[i] = 0;

}

ll Query(int i){

ll sum = 0;

while(i > 0){

sum += tree[i];

i -= ( i & -i );

}

return sum;

}

void Update(int i,ll val){

while(i <= N){

tree[i] += val;

i += (i & -i);

}

}

} FT;

int T[262144];

void update(int l,int r){

l += N;

r += N;

while(l<r){

if(l&1){

T[l++]++;

}

if(r&1){

T[--r]++;

}

l >>= 1;

r >>= 1;

}

}

int query(int x){

x += N;

int ans = 0;

while(x){

ans += T[x];

x >>= 1;

}

return ans;

}

void clear(int n){

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

}

int main(){

int t,n;cin>>t;

while(t--){

cin>>n;

N = 1;while(N < n+1) N<<=1;

FT.Clear();

clear(n+1);

ll num;

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

cin>>num;

FT.Update(i,num);

}

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

int lo=i,hi=n+1;

ll val = FT.Query(i)-FT.Query(i-1);

while((hi-lo)>1){

int mi = (hi+lo)/2;

ll suma = FT.Query(mi-1)-FT.Query(i);

if( suma > val) hi=mi;

else lo=mi;

}

update(i+1,hi);lo=0,hi=i;

while((hi-lo)>1){

int mi = (hi+lo)/2;

if(FT.Query(i)-FT.Query(mi)>2\*val) lo=mi;

else hi=mi;

}

update(hi,i);

}

for(int i=1;i<=n;i++) cout<<query(i)<<(char)(i==n?10:32);

}

}

**CRIBA OPTIMIZADA EN MEMORIA**

#include<bits/stdc++.h>

using namespace std;

const unsigned int N = (3e8);

const unsigned int M = (4e6);

unsigned int a,b,c,d,n;

unsigned int vis[M + 2];

unsigned int f(unsigned int x){

return (a\*x\*x\*x + b\*x\*x + c\*x + d);

}

bool prime(unsigned int x){

if(x==2 || x==3 || x==5) return true;

if(x%6!=1 && x%6!=5) return false;

x -= 6;

x /=3;

unsigned int pos = x/32;

unsigned int ter = x%32;

if(vis[pos] & (1<<ter)) return true;

return false;

}

void init(){

for(unsigned int i=0;i<=M;i++) vis[i] = 4294967295U;

for(unsigned int i=5;i\*i<=N;i++){

if(prime(i)){

for(unsigned int j=i\*i;j<=N;j+=i){

if(!prime(j)) continue;

unsigned int value = j - 6;

value /=3;

unsigned int pos = value/32;

unsigned int ter = value%32;

if(vis[pos] & (1<<ter)) vis[pos] ^= (1<<ter);

}

}

}

}

int main(){

init();

cin>>n>>a>>b>>c>>d;

unsigned int ans = 0;

for(int i=2;i<=n;i++){

if(!prime(i)) continue;

unsigned int cnt = 0,aux = n;

while(aux){

cnt += (aux/i);

aux /= i;

}

ans += cnt\*f(i);

}

cout<<ans<<'\n';

return 0;

}

**HOW MANY POT PERFECT ARE?**#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const ll INF = (1e18);

bool isSquare(ll x){

ll y = (ll) sqrt(x);

if(y\*y==x) return true;

if((y-1)\*(y-1)==x) return true;

if((y+1)\*(y+1)==x) return true;

return false;

}

bool prime(ll x){

for(ll i=2;i\*i<=x;i++){

if(x%i==0) return false;

}

return true;

}

ll f(ll x,ll y){

ll ans = 1;

for(int i=0;i<y;i++){

if(ans>INF/x) return INF+1;

ans\*=x;

}

return ans;

}

set<ll> used;

vector<ll> G;

void init(){

for(ll i=3;i<=64;i+=2){

for(ll j=2;;j++){

if(isSquare(j)) continue;

ll val = f(j,i);

if(val>INF) break;

if(used.find(val) != used.end()) continue;

G.push\_back(val);

used.insert(val);

}

}

sort(G.begin(),G.end());

}

int main(){

ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

init();

cout<<G.size()<<endl;

int t;cin>>t;

while(t--){

ll x;cin>>x;

ll ans = (ll) sqrt(x);

if(ans\*ans>x){

ans--;

}

ll p = lower\_bound(G.begin(),G.end(),x+1) - G.begin();

ans += p;

cout<<x-ans<<'\n';

}

return 0;

}

**KOSARAJU DAG COMPLETO**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = (2e5);

ll C[N+2];

vector<int> G[N+2],GG[N+2];

bool vis[N+2];

void addEdge(int x,int y){

if(x==y) return;

G[x].push\_back(y);

GG[y].push\_back(x);

}

stack<int> s;

int componente[N+5];

void dfs(int x){

vis[x] = 1;

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

if(vis[G[x][i]]) continue;

dfs(G[x][i]);

}

s.push(x);

}

void dfs2(int x,int id){

vis[x] = 1;

componente[x] = id;

for(int i=0;i<GG[x].size();i++){

if(vis[GG[x][i]]) continue;

dfs2(GG[x][i],id);

}

}

ll id=1;

vector<int> COND[N+5];//grafo condensado

int in[N+5];

vector<int> ACUM[N+5];

int n;

void kosaraju(){

memset(vis,0,sizeof(vis));

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

if(!vis[i]){

dfs(i);

}

}

memset(vis,0,sizeof(vis));

while(!s.empty()){

int val = s.top();

s.pop();

if(vis[val]) continue;

dfs2(val,id++);

}

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

int conden = componente[i];

ACUM[conden].push\_back(i);

}

set< pair<int,int> > M;

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

int componenteDeI = componente[i];

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

int componenteDeJ = componente[G[i][j]];

if(componenteDeI == componenteDeJ) continue;

if(M.find(make\_pair(componenteDeI,componenteDeJ))!=M.end()) continue;

COND[componenteDeI].push\_back(componenteDeJ);

M.insert(make\_pair(componenteDeI,componenteDeJ));

in[componenteDeI] ++;

}

}

vector<int> nodos;

ll ans = 0;

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

if(in[i]==0){//es nodo final

ll res = (1e6);

for(int j=0;j<ACUM[i].size();j++){

res = min(res,C[ACUM[i][j]]);

}

ans += res;

}

}

cout<<ans<<'\n';

}

int main(){

//ios\_base::sync\_with\_stdio(0);

cin>>n;

for(int i=1;i<=n;i++) cin>>C[i];

int num;

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

cin>>num;

addEdge(i,num);

}

kosaraju();

return 0;

}

**MEET IN THE MEEDLE (K-TH NUMERO FORMADO POR PRIMOS)**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const ll INF = (1e18);

void getval(int i,ll val,vector<ll> &v,vector<ll> &cont){

if(i==v.size()){

cont.push\_back(val);

return;

}

if(val<=INF/v[i])getval(i,val\*v[i],v,cont);

getval(i+1,val,v,cont);

}

vector<ll> valuesA,valuesB;

ll get(ll x){

ll ans = 0;

for(int i = 0, j = valuesA.size()-1; i < valuesB.size(); i++) {

if(valuesB[i] > x) break;

while(j >= 0 && valuesB[i] > x / valuesA[j]) j--;

ans += j+1ll;

}

return ans;

}

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

int n;cin>>n;

int p = n/2,r = n-p;

vector<ll> a(r),b(p);

for(int i=0;i<n/2;i++){

cin>>a[i]>>b[i];

}

if(r>p) cin>>a[r-1];

getval(0,1LL,a,valuesA);

getval(0,1LL,b,valuesB);

sort(valuesA.begin(),valuesA.end());

sort(valuesB.begin(),valuesB.end());

ll k;cin>>k;

ll lo=0,hi=INF+1;

while(hi-lo>1){

ll mi = (hi+lo)/2;

if(get(mi)<k) lo=mi;

else hi=mi;

}

cout<<hi<<'\n';

return 0;

}

**DINIC FLOW – PRIME FACTORS**

#include<bits/stdc++.h>

using namespace std;

#define pb push\_back

#define SZ(x) (int) x.size()

#define fore(i,a,b) for(int i=a,to=b;i<to;++i)

typedef long long ll;

const int N = (100);

const int MAXN = (5\*N+5);

const ll INF = (1e12);

int nodes,src,dst; // remember to init nodes

int dist[MAXN],q[MAXN],work[MAXN];

struct edge {int to,rev;ll f,cap;};

vector<edge> g[MAXN];

void add\_edge(int s, int t, ll cap){

g[s].pb((edge){t,SZ(g[t]),0,cap});

g[t].pb((edge){s,SZ(g[s])-1,0,0});

}

bool dinic\_bfs(){

fill(dist,dist+nodes,-1);dist[src]=0;

int qt=0;q[qt++]=src;

for(int qh=0;qh<qt;qh++){

int u=q[qh];

fore(i,0,SZ(g[u])){

edge &e=g[u][i];int v=g[u][i].to;

if(dist[v]<0&&e.f<e.cap)dist[v]=dist[u]+1,q[qt++]=v;

}

}

return dist[dst]>=0;

}

ll dinic\_dfs(int u, ll f){

if(u==dst)return f;

for(int &i=work[u];i<SZ(g[u]);i++){

edge &e=g[u][i];

if(e.cap<=e.f)continue;

int v=e.to;

if(dist[v]==dist[u]+1){

ll df=dinic\_dfs(v,min(f,e.cap-e.f));

if(df>0){e.f+=df;g[v][e.rev].f-=df;return df;}

}

}

return 0;

}

ll max\_flow(int \_src, int \_dst){

src=\_src;dst=\_dst;

ll result=0;

while(dinic\_bfs()){

fill(work, work+nodes, 0);

while(ll delta=dinic\_dfs(src,INF))result+=delta;

}

return result;

}

vector< pair<ll,ll> > pr[N+2];

void fact(ll x,int id){

for(ll i=2;i\*i<=x;i++){

ll cnt = 0;

while(x%i==0) cnt++,x/=i;

if(cnt) pr[id].pb({i,cnt});

}

if(x>1) pr[id].pb({x,1});

}

ll A[N+2];

ll init[N+2];

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

int n,m;

cin>>n>>m;

for(int i=1;i<=n;i++) cin>>A[i];

for(int i=1;i<=n;i++) fact(A[i],i);

src = 0;

nodes++;

for(int i=1;i<=n;i+=2){

init[i] = nodes;

for(int j=0;j<SZ(pr[i]);j++){

add\_edge(src,init[i]+j,pr[i][j].second);

nodes++;

}

}

for(int i=2;i<=n;i+=2){

init[i] = nodes;

for(int j=0;j<SZ(pr[i]);j++){

//add\_edge(src,init[i]+j,pr[i][j].second);

nodes++;

}

}

dst = nodes++;

for(int i=2;i<=n;i+=2){

for(int j=0;j<SZ(pr[i]);j++){

add\_edge(init[i]+j,dst,pr[i][j].second);

}

}

int a,b;

while(m--){

cin>>a>>b;

for(int i=0;i<SZ(pr[a]);i++){

for(int j=0;j<SZ(pr[b]);j++){

if(pr[a][i].first == pr[b][j].first){

if(a&1) add\_edge(init[a]+i,init[b]+j,INF);

else add\_edge(init[b]+j,init[a]+i,INF);

break;

}

}

}

}

cout<<max\_flow(src,dst)<<'\n';

return 0;

}

**DP DIGIT 2 VECES**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const ll MOD = (998244353LL);

ll memo[20][2][1100][2], memo2[20][2][1100][2];

string s;

int len,k;

void toString(ll x){

s.clear();

while(x){

s.push\_back(char('0'+x%10));

x/=10;

}

reverse(s.begin(),s.end());

}

ll dp(int pos,int menor,int mask,int init){

int p = \_\_builtin\_popcount(mask);

if(p>k) return 0LL;

if(pos==len) return 1LL;//acumulo el numero

if(memo[pos][menor][mask][init] != -1) return memo[pos][menor][mask][init];

ll &ans = memo[pos][menor][mask][init] = 0;

if(menor){

for(int i=0;i<10;i++){

if(init) ans += dp(pos+1,menor,mask|(1<<i),init);

else if(i>0) ans += dp(pos+1,menor,mask|(1<<i),1);

else ans += dp(pos+1,menor,mask,0);

ans %= MOD;

}

}else{

for(int i=0;i<=s[pos]-'0';i++){

if(init) ans += dp(pos+1,(i<s[pos]-'0'),mask|(1<<i),init);

else if(i>0) ans += dp(pos+1,(i<s[pos]-'0'),mask|(1<<i),1);

else ans += dp(pos+1,(i<s[pos]-'0'),mask,0);

ans %= MOD;

}

}

return ans;

}

ll pot[20];

ll cant(int pos,int menor,int mask,int init){

int p = \_\_builtin\_popcount(mask);

if(p>k) return 0LL;

if(pos==len) return 0LL;//no sumo nada >:v

if(memo2[pos][menor][mask][init] != -1) return memo2[pos][menor][mask][init];

ll &ans = memo2[pos][menor][mask][init] = 0;

if(menor){

for(int i=0;i<10;i++){

if(init){

ans += (i\*pot[len-pos-1]\*dp(pos+1,menor,mask|(1<<i),init));

ans += cant(pos+1,menor,mask|(1<<i),init);

}else if(i>0){

ans += (i\*pot[len-pos-1]\*dp(pos+1,menor,mask|(1<<i),1));

ans += cant(pos+1,menor,mask|(1<<i),1);

}else{

ans += (i\*pot[len-pos-1]\*dp(pos+1,menor,mask,0));

ans += cant(pos+1,menor,mask,0);

}

ans %= MOD;

}

}else{

for(int i=0;i<=s[pos]-'0';i++){

if(init){

ans += (i\*pot[len-pos-1]\*dp(pos+1,(i<s[pos]-'0'),mask|(1<<i),init));

ans += cant(pos+1,(i<s[pos]-'0'),mask|(1<<i),init);

}else if(i>0){

ans += (i\*pot[len-pos-1]\*dp(pos+1,(i<s[pos]-'0'),mask|(1<<i),1));

ans += cant(pos+1,(i<s[pos]-'0'),mask|(1<<i),1);

}else{

ans += (i\*pot[len-pos-1]\*dp(pos+1,(i<s[pos]-'0'),mask,0));

ans += cant(pos+1,(i<s[pos]-'0'),mask,0);

}

ans %= MOD;

}

}

return ans;

}

ll solve(ll up){

if(up==0) return 0;

memset(memo,-1,sizeof memo);

memset(memo2,-1,sizeof memo2);

toString(up);

len = s.size();

return cant(0,0,0,0);

}

int main(){

pot[0]=1LL;

for(int i=1;i<20;i++) pot[i]=(pot[i-1]\*10)%MOD;

ll l,r;cin>>l>>r>>k;

ll ans = solve(r) - solve(l-1);

ans = (ans%MOD + MOD)%MOD;

cout<<ans<<'\n';

return 0;

}

**SUBSTRING QUE SE REPITE MAS VECES Y ES GENERADO POR ALGUN ORDENAMIENTO DE OTRO STRING**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const ll MOD1 = (1e9+7);

const ll MOD2 = (1e9+9);

const int B = 29;

ll hpref[200005];

ll pot[200005];

ll hpref2[200005];

ll pot2[200005];

ll sumpref[200005];

ll multpref[200005];

ll multpref2[200005];

void init(){

pot[0] = 1;

for(int i=1;i<=200000;i++) pot[i]=(pot[i-1]\*B)%MOD1;

pot2[0] = 1;

for(int i=1;i<=200000;i++) pot2[i]=(pot2[i-1]\*B)%MOD2;

}

void getSumMult(string s){

memset(sumpref,0,sizeof sumpref);

memset(multpref,0,sizeof multpref);

memset(multpref2,0,sizeof multpref2);

sumpref[0] = (s[0] - 'a' + 1);

multpref[0] = (s[0] - 'a' + 1);

multpref2[0] = (s[0] - 'a' + 1);

for(int i=1;i<s.size();i++){

sumpref[i] = (sumpref[i-1] + (s[i]-'a'+1))%MOD1;

multpref[i] = (multpref[i-1] \* (s[i]-'a'+1))%MOD1;

multpref2[i] = (multpref2[i-1] \* (s[i]-'a'+1))%MOD2;

}

}

ll POT(ll x,ll y,ll mod){

if(y==0) return 1;

if(y==1) return x;

ll ans = 1;

if(y&1) ans = x;

ll val = POT(x,y/2,mod);

ans \*= val;

ans %= mod;

ans \*= val;

ans %= mod;

return ans;

}

ll inv(ll x,ll mod){

return POT(x,mod-2,mod);

}

ll subsum(int i,int j){

if(i==0) return sumpref[j];

return ((sumpref[j] - sumpref[i-1])%MOD1 + MOD1)%MOD1;

}

ll submult(int i,int j){

if(i==0) return multpref[j];

return (multpref[j] \* inv(multpref[i-1],MOD1))%MOD1;

}

ll submult2(int i,int j){

if(i==0) return multpref2[j];

return (multpref2[j] \* inv(multpref2[i-1],MOD2))%MOD2;

}

void getpref(string s){

memset(hpref,0,sizeof hpref);

hpref[0] = (s[0] - 'a' + 1);

for(int i=1;i<s.size();i++){

hpref[i] = (hpref[i-1]\*B + (s[i]-'a'+1))%MOD1;

}

memset(hpref2,0,sizeof hpref2);

hpref2[0] = (s[0] - 'a' + 1);

for(int i=1;i<s.size();i++){

hpref2[i] = (hpref2[i-1]\*B + (s[i]-'a'+1))%MOD2;

}

}

ll hsub(int i,int j){

if(i==0) return hpref[j];

return ((hpref[j] - hpref[i-1]\*pot[j-i+1])%MOD1 + MOD1)%MOD1;

}

ll hsub2(int i,int j){

if(i==0) return hpref2[j];

return ((hpref2[j] - hpref2[i-1]\*pot2[j-i+1])%MOD2 + MOD2)%MOD2;

}

map<pair<ll,ll>,int> M;

int ans[200005];

int main(){

init();

int t;cin>>t;

string a,b;

while(t--){

cin>>a>>b;

M.clear();

memset(ans,0,sizeof ans);

getpref(b);

getSumMult(b);

ll sum=0,mult=1,mult2=1;

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

sum += (a[i]-'a'+1);

mult \*= (a[i]-'a'+1);

mult %= MOD1;

mult2 \*= (a[i]-'a'+1);

mult2 %= MOD2;

}

for(int i=0;i<=b.size()-a.size();i++){

ll val1 = hsub(i,i+a.size()-1);

ll val2 = hsub2(i,i+a.size()-1);

if(subsum(i,i+a.size()-1)==sum && submult(i,i+a.size()-1)==mult && submult2(i,i+a.size()-1)==mult2){

pair<ll,ll> p = make\_pair(val1,val2);

if(M.count(p)){

ans[M[p]]++;

}else{

M[p] = i;

ans[i]++;

}

}

}

int maxi = 0;

vector<int> v;

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

if(ans[i] > maxi){

v.clear();

v.push\_back(i);

maxi = ans[i];

}else if(ans[i] == maxi){

v.push\_back(i);

}

}

if(maxi == 0) cout<<"-1\n";

else{

string res = b.substr(v[0],a.size());

for(int i=1;i<v.size();i++){

res = min(res,b.substr(v[i],a.size()));

}

cout<<res<<'\n';

}

}

return 0;

}

**COSTO MINIMO DE LIMPIAR UN CAMINO**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = (2e5);

const ll INF = (1e9);

ll LCA[N+2][25];

ll D[N+2][25];

int lvl[N+2];

ll G[N+2];

vector<int> GREV[N+2];

int n;

void dfs(int x,int level){

lvl[x] = level;

for(int i=0;i<GREV[x].size();i++) dfs(GREV[x][i],level+1);

}

void preprocess(){

for(int i=0;i<n;i++){

for(int j=0;(1<<j)<n;j++){

LCA[i][j] = -1;

D[i][j] = 0;

}

}

for(int i=0;i<n;i++){

LCA[i][0] = G[i];

D[i][0] = 1;

}

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

for(int i=0;i<n;i++){

if(LCA[i][j-1] != -1){

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

D[i][j] = D[i][j-1] + D[LCA[i][j-1]][j-1];

}

}

}

dfs(0,1);

}

void clear(){

for(int i=0;i<=n;i++){

GREV[i].clear();

G[i] = 0;

lvl[i] = 0;

}

}

int lca(int u,int v){

if(lvl[u] < lvl[v]) swap(u,v);

int lg = 31 - (\_\_builtin\_clz(lvl[u]));

for(int i=lg;i>=0;i--){

if(lvl[u] - (1<<i) >= lvl[v]){

u = LCA[u][i];

}

}

if(u==v) return u;

for(int i=lg;i>=0;i--){

if(LCA[u][i] != -1 && LCA[u][i] != LCA[v][i]){

u = LCA[u][i];

v = LCA[v][i];

}

}

return G[u];

}

set<int> S;//caminos borrados

ll dist(int pa,int hi){

if(pa==hi) return 0;

set<int> :: iterator it;

bool ok=1;

for(it=S.begin();it!=S.end();it++){

int p = (\*it);

if(p==pa) continue;

if(lca(hi,p)==p) ok=0;

}

if(!ok) return -INF;

int sube = lvl[hi] - lvl[pa];

ll ans = 0;

for(int i=0;i<25;i++){

if(sube & (1<<i)){

ans += D[hi][i];

hi = LCA[hi][i];

}

}

return ans;

}

vector<int> tree[N+2];

bool used[N+2];

void root(int x){

used[x] = 1;

for(int i=0;i<tree[x].size();i++){

int p = tree[x][i];

if(used[p]) continue;

GREV[x].push\_back(p);

G[p] = x;

root(p);

}

}

int main(){

ios::sync\_with\_stdio(0);

cin.tie(NULL);

cin>>n;

clear();

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

int a,b;

cin>>a>>b;

a--;b--;

tree[a].push\_back(b);

tree[b].push\_back(a);

}

root(0);

preprocess();

int q;cin>>q;

char type;

while(q--){

cin>>type;

int a,b;

cin>>a>>b;

a--;

b--;

if(type=='q'){//query

int ancestro = lca(a,b);

ll respuesta = dist(ancestro,a) + dist(ancestro,b);

if(respuesta<0) cout<<"Impossible\n";

else cout<<respuesta<<'\n';

}else if(type=='d'){//se destruye el camino de a,b

if(a==b) continue;

if(G[a]==b){

S.insert(a);

}else if(G[b]==a){

S.insert(b);

}

}else{

if(a==b) continue;

if(G[a]==b){

S.erase(a);

}else if(G[b]==a){

S.erase(b);

}

}

}

return 0;

}

**TRIE CON NUMEROS EN BINARIO**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = (1e5);

ll A[N+2];

int n;

ll k;

vector<int> base;

int pot = 30;

ll trie[N\*30 + 5][2];

int nodos = 1;

ll many[N\*60 + 5];

void addWord(vector<int> s){

int u = 0; //empezamos en la raíz

for(int i=0; i<pot; ++i){

int c = s[i];

if( trie[u][c] == 0) trie[u][c] = nodos++; //si no existe pref creamos nodo

u = trie[u][c];

many[u]++;

}

}

ll query(vector<int> rep){

int u = 0;

ll ans = 0;

for(int i=0;i<pot;i++){

int c = rep[i],d = base[i];

if(c==d){

if(c==1) ans += many[trie[u][1]];

if(trie[u][0]) u = trie[u][0];

else break;

}else{

if(c==0) ans += many[trie[u][0]];

if(trie[u][1]) u = trie[u][1];

else break;

}

}

return ans;

}

vector<int> f(ll x){

vector<int> v(pot,0);

int posi = 0;

while(x){

v[posi++] = x%2;

x/=2;

}

reverse(v.rbegin(),v.rend());

return v;

}

void clear(){

memset(A,0,sizeof A);

memset(trie,0,sizeof trie);

memset(many,0,sizeof many);

nodos = 1;

}

void solve(){

cin>>n>>k;

base = f(k);

for(int i=1;i<=n;i++) cin>>A[i];

for(int i=1;i<=n;i++) A[i]^=A[i-1];

ll ans = 0;

addWord(f(0LL));

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

ans += query(f(A[i]));

addWord(f(A[i]));

}

cout<<ans<<'\n';

clear();

}

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

int t;cin>>t;

while(t--) solve();

return 0;

}

**SQRT – AGREGAR ,QUITAR LINEAS, Y CONSULTAS**

#include<bits/stdc++.h>

using namespace std;

const int N = 100000;

const int BLOCK=200;

map<int,int> G[BLOCK+2];

int M[N+2];

void agregar(){

int k,b;

cin>>k>>b;

b%=k;

if(k>BLOCK){

while(b <= N){

M[b]++;

b+=k;

}

}else{

G[k][b]++;

}

}

void borrar(){

int k,b;cin>>k>>b;

b%=k;

if(k>BLOCK){

while(b <= N){

M[b]--;

b+=k;

}

}else{

G[k][b]--;

}

}

void query(){

int q;

cin>>q;

int ans = M[q];

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

int p = q%i;

if(G[i].count(p)) ans+=G[i][p];

}

cout<<ans<<'\n';

}

int main(){

ios::sync\_with\_stdio(0);

cin.tie(NULL);

int n;cin>>n;

char s;

for(int i=0;i<n;i++){

cin>>s;

if(s=='+') agregar();

else if(s=='-') borrar();

else query();

}

return 0;

}

**DIJKSTRA TREE**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = (3e5);

const ll INF = (1e16);

struct edge{

int to,id;

ll c;

edge(){}

edge(int \_to,ll \_c,int \_id){

to = \_to;

c = \_c;

id = \_id;

}

};

struct Node{

int u;ll w;

Node(){}

Node(int \_u,ll \_w){

u = \_u;

w = \_w;

}

};

bool arb[N+2];

ll D[N+2];

int last[N+2];

int n;

bool operator <(const Node &a,const Node &b){

return a.w>b.w;

}

vector<edge> G[N+2];

ll costos[N+2];

void dijkstra(int src){

fill(D+1,D+n+1,INF);

D[src] = 0;

priority\_queue<Node> Q;

Q.push(Node(src,0));

while(!Q.empty()){

Node a = Q.top();

Q.pop();

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

edge &cur = G[a.u][i];

ll cost = cur.c;

ll dst = cur.to;

int id = cur.id;

if(D[dst] > cost + D[a.u]){

D[dst] = cost+D[a.u];

arb[last[dst]] = 0;

arb[id] = 1;

last[dst] = id;

Q.push(Node(dst,D[dst]));

}else if((D[dst] == cost + D[a.u]) && costos[last[dst]]>cost){

arb[last[dst]] = 0;

arb[id] = 1;

last[dst] = id;

}

}

}

}

int main(){

ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

int m;cin>>n>>m;

int a,b;ll c;

costos[0] = INF;

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

cin>>a>>b>>c;

G[a].push\_back(edge(b,c,i));

G[b].push\_back(edge(a,c,i));

costos[i] = c;

}

int src;cin>>src;

dijkstra(src);

ll suma = 0;

vector<int> arbol;

for(int i=1;i<=m;i++) if(arb[i]) arbol.push\_back(i);

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

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

edge &cur = G[i][j];

if(arb[cur.id]) suma+=cur.c;

}

}

suma /= 2;

cout<<suma<<"\n";

for(int i=0;i<arbol.size();i++) cout<<arbol[i]<<(char)(i+1==arbol.size()?10:32);

return 0;

}

**DP – NUMERO DE SUBARRAYS DONDE SE PUEDE CONSEGUIR SUMA 0**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = (1e3);

const ll MOD = (1e9+7);

ll memo[N+2][20\*N+5];

ll A[N+2];

int n;

ll dp(int pos,ll sum){

if(memo[pos][sum+10\*N] != -1) return memo[pos][sum+10\*N];

ll &ans = memo[pos][sum+10\*N] = (sum==0);

if(pos>n) return ans;

ans += dp(pos+1,sum+A[pos]);

ans %= MOD;

ans += dp(pos+1,sum-A[pos]);

ans %= MOD;

return ans;

}

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

cin>>n;

for(int i=1;i<=n;i++) cin>>A[i];

memset(memo,-1,sizeof memo);

ll ans = 0;

for(int i=n;i>=1;i--) ans+=(dp(i,0)-1),ans%=MOD;

cout<<ans<<'\n';

return 0;

}

**BELLMON FORD – INECUACIONES**

#include<bits/stdc++.h>

#define fore(i,a,b) for(int i=a,to=b;i<to;++i)

#define fst first

#define snd second

using namespace std;

typedef long long ll;

const ll INF = (1e18);

const int MAXN = 100;

int n,m;

vector<pair<int,ll> > g[MAXN+5]; // u->[(v,cost)]

ll dist[MAXN+5];

bool bford(int src){ // O(nm)

fill(dist,dist+n,INF);dist[src]=0;

fore(\_,0,n-1)fore(x,0,n)if(dist[x]!=INF)for(auto t:g[x]){

dist[t.fst]=min(dist[t.fst],dist[x]+t.snd);

}

fore(x,0,n)if(dist[x]!=INF)for(auto t:g[x]){

if(dist[t.fst]>dist[x]+t.snd){

return true;

}

}

return false;

}

void clear(){

for(int i=0;i<=n+1;i++) g[i].clear();

}

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

while(cin>>n){

if(n==0) break;

clear();

cin>>m;

int a,b,c;string st;

for(int i=0;i<m;i++){

cin>>a>>b>>st>>c;

b+=a;

a--;

if(st=="gt") g[b].push\_back({a,-c-1});

else g[a].push\_back({b,c-1});

}

for(int i=0;i<=n;i++){

g[n+1].push\_back({i,0});

}

n+=2;

if(bford(n-1)){

cout<<"successful conspiracy\n";

}else{

cout<<"lamentable kingdom\n";

}

}

return 0;

}

**BELLMON FORD – MAXIMUN AVERAGE CICLE**

#include<bits/stdc++.h>

#define fore(i,a,b) for(int i=a,ThxDem=b;i<ThxDem;++i)

#define fst first

#define snd second

using namespace std;

typedef long long ll;

typedef long double ld;

const ld INF = (1e7);

const ld EPS = (1e-6);

const int MAXN = (50);

int n,m;

vector<pair<int,ld> > g[MAXN+2]; // u->[(v,cost)]

ld dist[MAXN+2];

bool bford(int src,ld search){ // O(nm)

fill(dist,dist+n,INF);dist[src]=0;

fore(\_,0,n-1)fore(x,0,n)if(abs(dist[x]-INF)>EPS)for(auto t:g[x]){

dist[t.fst]=min(dist[t.fst],dist[x]+t.snd-search);

}

fore(x,0,n)if(abs(dist[x]-INF)>EPS)for(auto t:g[x]){

if(dist[t.fst]>dist[x]+t.snd-search){

return true;

}

}

return false;

}

int caso = 0;

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

int t;cin>>t;

while(t--){

cin>>n>>m;

for(int i=0;i<=n;i++) g[i].clear();

int a,b;ll c;

for(int i=0;i<m;i++){

cin>>a>>b>>c;

g[a].push\_back(make\_pair(b,(ld)c));

}

for(int i=1;i<=n;i++) g[0].push\_back(make\_pair(i,0.0L));

n++;

ld lo=0,hi=INF;

bool ok=0;

while((hi-lo)>EPS){

ld mi = (hi+lo)/2.0L;

bool negCiclo=bford(0,mi);

if(negCiclo) ok=1,hi=mi;

else lo=mi;

}

if(!ok) printf("Case #%d: No cycle found.\n",++caso);

else printf("Case #%d: %.2lf\n",++caso,(double)hi);

}

return 0;

}

**USANSO KARP EN LUGAR DE BELLMO FORD**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int INF = 10000;

const int N = 1000;

struct edge{

int v;

ll w;

edge(){} edge(int v, int w) : v(v), w(w) {}

};

map<string,int> cd;

vector<edge> g[N+2];

long long d[N+2][N+2];

int karp(int n){

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

if(!g[i].empty())

g[n].push\_back(edge(i,0));

n++;

for(int i = 0;i<n;++i) fill(d[i],d[i]+(n+1),INT\_MAX);

d[n-1][0] = 0;

for (int k = 1;k<=n;++k) for (int u = 0;u<n;++u){

if(d[u][k-1]==INT\_MAX) continue;

for(int i = g[u].size()-1;i>=0;--i) d[g[u][i].v][k] = min(d[g[u][i].v][k],d[u][k-1]+g[u][i].w);

}

bool flag = true;

for(int i = 0;i<n;++i) if(d[i][n]!=INT\_MAX) flag = false;

if(flag) return true;

double ans = 1e15;

for(int u = 0;u+1<n;++u){

if(d[u][n]==INT\_MAX) continue;

double W = -1e15;

for(int k = 0;k<n;++k) if(d[u][k]!=INT\_MAX) W = max(W,(double)(d[u][n]-d[u][k])/(n-k));

ans = min(ans,W);

}

ans = -ans;

cout<<ans<<'\n';

return false;

}

int main() {

ios\_base::sync\_with\_stdio(0);cin.tie(NULL);

string cur(2, '0');

for (int i = 0; i < 26; ++i) {

for (int j = 0; j < 26; ++j) {

cur[0] = char(i + 'a');

cur[1] = char(j + 'a');

cd[cur] = i \* 26 + j;

}

}

int n;

while (cin>>n) {

if(n==0) break;

string line;

for (int i = 0; i < N; ++i) {

g[i].clear();

}

for (int i = 0; i < n; ++i) {

cin >> line;

if (line.size() < 2) continue;

int u = cd[line.substr(0, 2)];

int v = cd[line.substr(line.size() - 2, 2)];

g[u].push\_back(edge(v, -line.size()));

}

if (karp(cd.size())) {

cout << "No solution." << endl;

}

}

return 0;

}

**FLOW – LATIN AMERICA ICPC 2015**

#include<bits/stdc++.h>

using namespace std;

#define SZ(a) (int)a.size()

#define pb push\_back

#define fore(i,a,b) for(int i=a,to=b;i<to;i++)

#define fi first

#define snd second

typedef long long ll;

const int N = (50);

const int MAXN = (2\*N+10);

const ll INF = (1e12);

struct edge {int to,rev;ll f,cap;};

struct Dinic{

int nodes,src,dst;

int dist[MAXN],q[MAXN],work[MAXN];

vector<edge> g[MAXN];

Dinic(int \_nodes,int \_src,int \_dst){

nodes = \_nodes;src = \_src;dst = \_dst;

}

void add\_edge(int s, int t, ll cap){

g[s].pb((edge){t,SZ(g[t]),0,cap});

g[t].pb((edge){s,SZ(g[s])-1,0,0});

}

bool dinic\_bfs(){

fill(dist,dist+nodes,-1);dist[src]=0;

int qt=0;q[qt++]=src;

for(int qh=0;qh<qt;qh++){

int u=q[qh];

fore(i,0,SZ(g[u])){

edge &e=g[u][i];int v=g[u][i].to;

if(dist[v]<0&&e.f<e.cap)dist[v]=dist[u]+1,q[qt++]=v;

}

}

return dist[dst]>=0;

}

ll dinic\_dfs(int u, ll f){

if(u==dst)return f;

for(int &i=work[u];i<SZ(g[u]);i++){

edge &e=g[u][i];

if(e.cap<=e.f)continue;

int v=e.to;

if(dist[v]==dist[u]+1){

ll df=dinic\_dfs(v,min(f,e.cap-e.f));

if(df>0){e.f+=df;g[v][e.rev].f-=df;return df;}

}

}

return 0;

}

ll max\_flow(){

ll result=0;

while(dinic\_bfs()){

fill(work, work+nodes, 0);

while(ll delta=dinic\_dfs(src,INF))result+=delta;

}

return result;

}

};

bool vis[N+5];

void solve(int n){

int ans = 0;

vector< pair<int,int> > v(n);

for(int i=0;i<n;i++) cin>>v[i].fi>>v[i].snd;

for(int i=0;i<n;i++){

int src = 0,dst = 2\*n+1;

Dinic dinic(2\*n+2,src,dst);

memset(vis,0,sizeof vis);

int votos = 0;

for(int j=0;j<n;j++) if(i!=j) dinic.add\_edge(src,j+1,1);

for(int j=0;j<n;j++){

if(j==i){

vis[v[j].fi]=1;

vis[v[j].snd]=1;

}else if(v[j].fi==i+1 || v[j].snd==i+1) votos++;

else{

dinic.add\_edge(j+1,v[j].fi+n,1);

dinic.add\_edge(j+1,v[j].snd+n,1);

}

}

if(votos<=1){

ans++;

continue;

}

for(int j=0;j<n;j++){

if(i==j) continue;

if(vis[j+1]) dinic.add\_edge(j+1+n,dst,votos-2);

else dinic.add\_edge(j+1+n,dst,votos-1);

}

int faltan = n-votos-1;

int flow = dinic.max\_flow();

if(flow < faltan) ans++;

}

cout<<ans<<'\n';

}

int main(){

int n;

while(cin>>n) solve(n);

return 0;

}

**COMPARANDO 2 POLIGONOS, ROTADOS, TRASLADADOS, Y EXPANDIDO/CONTRAIDO**

#include<bits/stdc++.h>

using namespace std;

#define Vector Point

typedef long long ll;

typedef long double ld;

const ll MOD2 = (1e9+9);

const ll MOD1 = (1e9+7);

const ld EPS = (1e-9);

struct Point{

ll x,y;

Point(){}

Point(ll \_x,ll \_y){

x = \_x;

y = \_y;

}

ll mod2(){

return (x\*x+y\*y);

}

};

Point operator +(const Point &a ,const Point &b){

return Point(a.x+b.x,a.y+b.y);

}

Point operator -(const Point &a ,const Point &b){

return Point(a.x-b.x,a.y-b.y);

}

Point operator \*(const Point &a,ll k){

return Point(a.x\*k,a.y\*k);

}

bool operator <(const Point &a, const Point &b){

if(a.x != b.x) return a.x < b.x;

return a.y < b.y;

}

ll cross(const Vector &A, const Vector &B){

return A.x \* B.y - A.y \* B.x;

}

ll area(const Point &A, const Point &B, const Point &C) {

return cross(B - A, C - A);

}

vector< pair< ll, pair<ll,ll> > > f(vector<Point> b,ll multi){

int len = b.size();

vector< pair< ll, pair<ll,ll> > > ans;

for(int i=0;i<len;i++){

Point uno = b[i],dos = b[(i+1)%len],tres = b[(i-1+len)%len];

ans.push\_back(make\_pair((dos-uno).mod2()\*multi-(tres-uno).mod2()\*multi,make\_pair((dos-uno).mod2()\*multi+(tres-uno).mod2()\*multi,(tres-dos).mod2()\*multi)));

}

return ans;

}

vector< ll > hashSum(vector< pair< ll, pair<ll,ll> > > v,ll mod){

vector< ll > ans;

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

v[i].first %= mod;

v[i].first += mod;

v[i].first %= mod;

v[i].second.first %= mod;

v[i].second.first += mod;

v[i].second.first %= mod;

v[i].second.second %= mod;

v[i].second.second += mod;

v[i].second.second %= mod;

}

for(int i=0;i<v.size();i++) ans.push\_back((v[i].first + v[i].second.first + v[i].second.second)%mod);

return ans;

}

vector< ll > hashMul(vector< pair< ll, pair<ll,ll> > > v,ll mod){

vector< ll > ans;

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

v[i].first %= mod;

v[i].first += mod;

v[i].first %= mod;

v[i].second.first %= mod;

v[i].second.first += mod;

v[i].second.first %= mod;

v[i].second.second %= mod;

v[i].second.second += mod;

v[i].second.second %= mod;

}

for(int i=0;i<v.size();i++) ans.push\_back(((v[i].first\*v[i].second.first)%mod\*v[i].second.second)%mod);

return ans;

}

vector<int> KMP(vector<ll> S,vector<ll> K){

vector<int> T(K.size() + 1, -1);

for(int i = 1; i <= K.size(); i++){

int pos = T[i - 1];

while(pos != -1 && K[pos] != K[i - 1]) pos = T[pos];

T[i] = pos + 1;

}

vector<int> matches;

for(int sp = 0, kp = 0; sp < S.size(); sp++){

while(kp != -1 && (kp == K.size() || (K[kp] != S[sp]) /\*abs(S[sp]\*base - K[kp])>EPS\*/ ))

kp = T[kp];

kp++;

if(kp == K.size()) matches.push\_back(sp + 1 - K.size());

}

return matches;

}

int vis[400005];

ld base,baseCua;

bool eq(vector< pair< ll, pair<ll,ll> > > v1,vector< pair< ll, pair<ll,ll> > > v2){

int len = v1.size();

vector< ll > hashSum1 = hashSum(v1,MOD1);

vector< ll > hashSum2 = hashSum(v2,MOD1);

vector< ll > hashMul1 = hashMul(v1,MOD1);

vector< ll > hashMul2 = hashMul(v2,MOD1);

//duplico el 2do

for(int i=0;i<len;i++) hashSum2.push\_back(hashSum2[i]);

for(int i=0;i<len;i++) hashMul2.push\_back(hashMul2[i]);

vector<int> kmp1 = KMP(hashSum2,hashSum1), kmp2 = KMP(hashMul2,hashMul1);

for(int i=0;i<kmp1.size();i++) vis[kmp1[i]]++;

for(int i=0;i<kmp2.size();i++) vis[kmp2[i]]++;

vector< ll > hashSum3 = hashSum(v1,MOD2);

vector< ll > hashSum4 = hashSum(v2,MOD2);

vector< ll > hashMul3 = hashMul(v1,MOD2);

vector< ll > hashMul4 = hashMul(v2,MOD2);

//duplico el 2do

for(int i=0;i<len;i++) hashSum4.push\_back(hashSum4[i]);

for(int i=0;i<len;i++) hashMul4.push\_back(hashMul4[i]);

vector<int> kmp3 = KMP(hashSum4,hashSum3), kmp4 = KMP(hashMul4,hashMul3);

for(int i=0;i<kmp3.size();i++) vis[kmp3[i]]++;

for(int i=0;i<kmp4.size();i++) vis[kmp4[i]]++;

for(int i=0;i<=3\*len+3;i++){

if(vis[i]==4) return true;

}

return false;

}

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

int n;cin>>n;

vector<Point> v1(n),v2(n);

for(int i=0;i<n;i++) cin>>v1[i].x>>v1[i].y;

for(int i=0;i<n;i++) cin>>v2[i].x>>v2[i].y;

ll area1 = 0;

for(int i=1;i<n-1;i++) area1+=area(v1[0],v1[i],v1[i+1]);

if(area1 < 0){

area1 = -area1;

for(int i=0;i<n/2;i++){

swap(v1[i],v1[n-i-1]);

}

}

ll area2 = 0;

for(int i=1;i<n-1;i++) area2+=area(v2[0],v2[i],v2[i+1]);

if(area2 < 0){

area2 = -area2;

for(int i=0;i<n/2;i++){

swap(v2[i],v2[n-i-1]);

}

}

ll gcd = \_\_gcd(area1,area2);

area1 /= gcd;

area2 /= gcd;

vector< pair< ll, pair<ll,ll> > > cmp1 = f(v1,area2);

vector< pair< ll, pair<ll,ll> > > cmp2 = f(v2,area1);

if(eq(cmp1,cmp2)) cout<<"MISMO\n";

else cout<<"OTRO\n";

return 0;

}

**POLIMONIO, CALCULANDO EN CUANTAS BASES SE CUMPLE UNA ECUACION**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

#define fore(i,a,b) for(int i=a,to=b;i<to;++i)

#define pb(x) push\_back(x)

typedef ll tp; // type of polynomial

template<class T=tp>

struct poly { // poly<> : 1 variable, poly<poly<>>: 2 variables, etc.

vector<T> c;

T& operator[](ll k){return c[k];}

poly(vector<T>& c):c(c){}

poly(ll k):c(k){}

poly(){}

poly operator+(poly<T> o){

int m=c.size(),n=o.c.size();

poly res(max(m,n));

fore(i,0,m)res[i]=res[i]+c[i];

fore(i,0,n)res[i]=res[i]+o.c[i];

return res;

}

poly operator\*(tp k){

poly res(c.size());

fore(i,0,c.size())res[i]=c[i]\*k;

return res;

}

poly operator\*(poly o){

int m=c.size(),n=o.c.size();

poly res(m+n-1);

fore(i,0,m)fore(j,0,n)res[i+j]=res[i+j]+c[i]\*o.c[j];

return res;

}

poly operator-(poly<T> o){return \*this+(o\*-1);}

T operator()(tp v){

T sum(0);

for(int i=c.size()-1;i>=0;--i)sum=sum\*v+c[i];

return sum;

}

bool isConstant(){

for(int i=1;i<c.size();i++){

if(c[i]!=0) return false;

}

return true;

}

};

// example: p(x,y)=2\*x^2+3\*x\*y-y+4

// poly<poly<>> p={{4,-1},{0,3},{2}}

// printf("%d\n",p(2)(3)) // 27 (p(2,3))

set<tp> roots(poly<> p){ // only for integer polynomials

set<tp> r;

while(!p.c.empty()&&!p.c.back())p.c.pop\_back();

if(!p(0))r.insert(0);

if(p.c.empty())return r;

tp a0=0,an=abs(p[p.c.size()-1]);

for(int k=0;!a0;a0=abs(p[k++]));

vector<tp> ps,qs;

fore(i,1,sqrt(a0)+1)if(a0%i==0)ps.pb(i),ps.pb(a0/i);

fore(i,1,sqrt(an)+1)if(an%i==0)qs.pb(i),qs.pb(an/i);

for(auto pt:ps)for(auto qt:qs)if(pt%qt==0){

tp x=pt/qt;

if(!p(x))r.insert(x);

if(!p(-x))r.insert(-x);

}

return r;

}

vector<string> sum(string x){

x+="+";

string base="";

vector<string> ans;

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

if(x[i]=='+'){

ans.push\_back(base);

base="";

}else{

base.push\_back(x[i]);

}

}

return ans;

}

vector<tp> iden;

vector<tp> vacio;

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

/\*vector<tp> v(5);

v[0] = 60;

v[1] = 44;

v[2] = 45;

v[3] = -15;

v[4] = 1;

poly<> p(v);

set<tp> solve = roots(p);

set<tp> :: iterator it;

for(it=solve.begin();it!=solve.end();it++){

cout<<(\*it)<<endl;

}\*/

string base;

iden.pb(1);

vacio.pb(0);

while(cin>>base){

if(base=="=") break;

int maxi = 1;

int pos;

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

if(base[i]=='=') pos=i;

if(base[i]>='0' && base[i]<='9') maxi = max(maxi,base[i]-'0');

}

string term1 = base.substr(0,pos);

string term2 = base.substr(pos+1,base.size());

vector<string> sumas1 = sum(term1);

vector<string> sumas2 = sum(term2);

poly<> comp1(vacio);

for(int i=0;i<sumas1.size();i++){

poly<> cur(iden);

sumas1[i]+="\*";

vector<tp> mult;

for(int j=0;j<sumas1[i].size();j++){

if(sumas1[i][j]=='\*'){

reverse(mult.begin(),mult.end());

poly<> multiplicando(mult);

cur=cur\*multiplicando;

mult.clear();

}else mult.pb(sumas1[i][j]-'0');

}

comp1 = cur+comp1;

}

poly<> comp2(vacio);

for(int i=0;i<sumas2.size();i++){

poly<> cur(iden);

sumas2[i]+="\*";

vector<tp> mult;

for(int j=0;j<sumas2[i].size();j++){

if(sumas2[i][j]=='\*'){

reverse(mult.begin(),mult.end());

poly<> multiplicando(mult);

cur=cur\*multiplicando;

mult.clear();

}else mult.pb(sumas2[i][j]-'0');

}

comp2 = cur+comp2;

}

comp1 = comp2 - comp1;

if(comp1.isConstant()){

if(comp1.c[0]!=0){

cout<<"\*"<<'\n';

}else{

cout<<maxi+1<<"+\n";

}

}else{

set<tp> res = roots(comp1);

set<tp> :: iterator it;

vector<tp> respu;

for(it=res.begin();it!=res.end();it++){

tp p = (\*it);

if(p<=maxi) continue;

respu.pb(p);

}

if(respu.size()==0) cout<<"\*\n";

else{

for(int i=0;i<respu.size();i++) cout<<respu[i]<<(char)(i+1==respu.size()?10:32);

}

}

}

return 0;

}

**MINIMA DISTANCIA ENTRE 2 PUNTOS PASANDO POR UN LADO DE UN POLIGONO**

#include<bits/stdc++.h>

using namespace std;

#define Vector pt

#define pb push\_back

typedef long long ll;

typedef long double ld;

const ld DINF = (1e200);

const ld EPS = (1e-9);

struct pt { // for 3D add z coordinate

ld x,y;

pt(ld x, ld y):x(x),y(y){}

pt(){}

ld norm2(){return \*this\*\*this;}

ld norm(){return sqrt(norm2());}

bool operator==(pt p){return abs(x-p.x)<EPS&&abs(y-p.y)<EPS;}

pt operator+(pt p){return pt(x+p.x,y+p.y);}

pt operator-(pt p){return pt(x-p.x,y-p.y);}

pt operator\*(double t){return pt(x\*t,y\*t);}

pt operator/(double t){return pt(x/t,y/t);}

ld operator\*(pt p){return x\*p.x+y\*p.y;}

ld angle(pt p){ // redefine acos for values out of range

return acos(\*this\*p/(norm()\*p.norm()));}

pt unit(){return \*this/norm();}

ld operator%(pt p){return x\*p.y-y\*p.x;}

// 2D from now on

};

struct ln {

pt p,pq;

ln(pt p, pt q):p(p),pq(q-p){}

ln(){}

bool has(pt r){return dist(r)<EPS;}

bool seghas(pt r){return has(r)&&(r-p)\*(r-(p+pq))-EPS<0;}

bool operator/(ln l){return abs(pq.unit()%l.pq.unit())<EPS;} // 2D

bool operator==(ln l){return \*this/l&&has(l.p);}

pt operator^(ln l){ // intersection

if(\*this/l)return pt(DINF,DINF);

pt r=l.p+l.pq\*((p-l.p)%pq/(l.pq%pq));

return r;

}

pt proj(pt r){return p+pq\*((r-p)\*pq/pq.norm2());}

pt ref(pt r){

if(seghas(r)) return r;

return proj(r)\*2-r;

}

double dist(pt r){return (r-proj(r)).norm();}

};

int caso;

void solve(){

int n;

cin>>n;

vector<pt> v;

vector<ln> w;

pt a,b;

for(int i=0;i<n;i++) cin>>a.x>>a.y,v.pb(a);

for(int i=0;i<n;i++) w.pb(ln(v[i],v[(i+1)%n]));

int q;cin>>q;

printf("Case %d:\n",++caso);

while(q--){

cin>>a.x>>a.y>>b.x>>b.y;

ld dist=DINF;pt ans;

for(int i=0;i<n;i++){

pt op = w[i].ref(b);

ld curDist = (op-a).norm();

if(a==op){

dist = curDist;ans = a;continue;

}

if(w[i]/(ln(a,op))) continue;

pt curPoint = w[i]^ln(a,op);

if(curDist<=dist+EPS){

dist = curDist;

ans = curPoint;

}

}

printf("%.7f %.7f %.7f\n",(double)dist,(double)ans.x,(double)ans.y);

}

}

int main(){

//freopen ("flags.in","r",stdin);

int t;cin>>t;

while(t--) solve();

return 0;

}

**ORDERED SET C++11**

#include<bits/stdc++.h>

#include <ext/pb\_ds/assoc\_container.hpp>

#include <ext/pb\_ds/tree\_policy.hpp>

using namespace std;

#pragma GCC optimize ("O3")

#pragma GCC optimize ("unroll-loops")

#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,tune=native")

using namespace \_\_gnu\_pbds;

typedef long long ll;

typedef tree<ll,null\_type,greater<ll>,rb\_tree\_tag,tree\_order\_statistics\_node\_update> ordered\_set;

ll read(int n){

ordered\_set X;

ll num;

ll ans = 0;

vector< pair<ll,ll> > v(n);

for(int i=0;i<n;i++){

cin>>v[i].first>>v[i].second;

}

sort(v.begin(),v.end());

for(int i=0;i<n;i++){

//X.insert(v[i].second);

ans += (X.order\_of\_key(v[i].second));

X.insert(v[i].second);

}

return ans;

}

int main(){

ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

ll x,y;cin>>x>>y;

ll n,m;

cin>>n>>m;

ll ans = (n+1)\*(m+1);

ans += read(n);

ans += read(m);

cout<<ans<<'\n';

return 0;

}

**MINCUT EN UN GRAFO NO DIRIGIDO(WAGNER)**

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const ll INF = (1e12);

const int N = 50;

ll g[N+2][N+2];

ll dist[N+2];

bool vis[N+2];

void addEdge(int u, int v, ll c){

g[u][v] += c;

g[v][u] += c;

}

ll Wagner(vector<int> &v){//vertices

ll mincut = INF;

while(v.size() > 1){

int u = v[0];

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

vis[v[i]] = false;

dist[v[i]] = g[u][v[i]];

}

vis[u] = true;

for(int t=0;t<v.size()-2;t++){

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

if (!vis[v[i]]){

if(vis[u] || dist[v[i]] > dist[u]) u = v[i];

}

}

vis[u] = true;

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

if (!vis[v[i]]) dist[v[i]]+=g[u][v[i]];

}

}

int t = -1;

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

if (!vis[v[i]]) t = v[i];

}

mincut = min(mincut, dist[t]);

v.erase(find(v.begin(),v.end(),t));

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

addEdge(u, v[i], g[v[i]][t]);

}

}

return mincut;

}

int main(){

int n, m;cin>>n>>m;

ll tot = 0;

for(int i=0;i<m;i++){

int k, f, u;

cin>>k>>f;

vector<int> group;

for(int j=0;j<k;j++){

cin>>u;u--;

group.push\_back(u);

}

tot += 2\*f;

if (k == 2) addEdge(group[0], group[1], 2 \* f);

else{

addEdge(group[0], group[1], f);

addEdge(group[1], group[2], f);

addEdge(group[2], group[0], f);

}

}

vector<int> vertices;

for(int i=0;i<n;i++) vertices.push\_back(i);

ll mincut = Wagner(vertices);

cout<<(tot - mincut)/2<<'\n';

return 0;

}

**DETERMINANTE DE UNA MATRIX**

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define fst first

#define snd second

#define fore(i,a,b) for(int i=a,to=b;i<to;++i)

using namespace std;

typedef long long ll;

typedef long double ld;

const double EPS = (1e-9);

double reduce(vector<vector<double> >& x){ // returns determinant

int n=x.size(),m=x[0].size();

int i=0,j=0;double r=1.0;

while(i<n&&j<m){

int l=i;

fore(k,i+1,n)if(abs(x[k][j])>abs(x[l][j]))l=k;

if(abs(x[l][j])<EPS){j++;r=0.0;continue;}

if(l!=i){r=-r;swap(x[i],x[l]);}

r\*=x[i][j];

for(int k=m-1;k>=j;k--)x[i][k]/=x[i][j];

fore(k,0,n){

if(k==i)continue;

for(int l=m-1;l>=j;l--)x[k][l]-=x[k][j]\*x[i][l];

}

i++;j++;

}

return r;

}

int main(){

int n;

while(cin>>n){

if(n==0) break;

vector<vector<double> > x(n,vector<double>(n,0.0));

fore(i,0,n)fore(j,0,n) cin>>x[i][j];

cout<<(ll)round(reduce(x))<<'\n';

}

puts("\*");

return 0;

}

**CRAMMER – ECUACION DE UN PLANO CON 3 PUNTOS**

#include<bits/stdc++.h>

using namespace std;

#define Vector Point

typedef long long ll;

struct Point{

ll x,y,z;

Point(){}

Point(ll \_x,ll \_y,ll \_z){

x = \_x;

y = \_y;

z = \_z;

}

};

struct Mat{

ll M[3][3];

Mat(){

memset(M,0,sizeof M);

}

Mat(Point a,Point b,Point c){

M[0][0] = a.x;M[1][0] = b.x;M[2][0] = c.x;

M[0][1] = a.y;M[1][1] = b.y;M[2][1] = c.y;

M[0][2] = a.z;M[1][2] = b.z;M[2][2] = c.z;

}

void set(int x){

for(int i=0;i<3;i++){

M[i][x] = 1LL;

}

}

ll det(){

ll ans = 0;

for(int i=0;i<3;i++){

ll cur1 = 1,cur2 = 1;

for(int j=0;j<3;j++){

cur1 \*= M[j][(i+j)%3];

cur2 \*= M[j][(i-j+3)%3];

}

ans += (cur1-cur2);

}

return ans;

}

};

vector< Point > v;

Point operator -(const Point &a,const Point &b){

return Point(a.x-b.x,a.y-b.y,a.z-b.z);

}

bool notline(Point a,Point b,Point c){

a=a-c;

b=b-c;

if ( a.y\*b.z==b.y\*a.z && a.x\*b.z==b.x\*a.z && a.x\*b.y==b.x\*a.y ) return false ;

return true ;

}

Vector Crammer(Point a,Point b,Point c){

Mat matriz(a,b,c);

Mat matrizX = matriz;

matrizX.set(0);

Mat matrizY = matriz;

matrizY.set(1);

Mat matrizZ = matriz;

matrizZ.set(2);

Point ec;

ec.x = matrizX.det();

ec.y = matrizY.det();

ec.z = matrizZ.det();

return ec;

}

ll eval(Vector a,Point c){

return c.x\*a.x + c.y\*a.y + c.z\*a.z;

}

int main(){

int n;cin>>n;

if(n<=3){

cout<<"TAK\n";

return 0;

}

v.resize(n);

ll a,b,c;

for(int i=0;i<n;i++){

cin>>a>>b>>c ;

v[i] = Point(a,b,c);

}

Point p1=v[0],p2=v[1],p3;

bool ok = 0;

for(int i=2;i<n;i++){

if(notline(p1,p2,v[i])){

ok = 1;

p3 = v[i];

break;

}

}

if(!ok){

cout<<"TAK\n";

return 0;

}

Mat matriz(p1,p2,p3);

ll determ = matriz.det();

Vector ec = Crammer(p1,p2,p3);

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

ll val = eval(ec,v[i]);

if(val != determ){

cout<<"NIE\n";

return 0;

}

}

cout<<"TAK\n";

return 0;

}

**EXPONENCIACION DE MATRICES**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const ll MOD=(1e9 + 7);

const int K = 4;//numero de recursiones

struct Matriz{

ll M[K][K];

Matriz(){

for(int i=0;i<K;i++) for(int j=0;j<K;j++) M[i][j] = 0LL;

}

void iden(){

for(int i=0;i<K;i++) for(int j=0;j<K;j++) if(i==j) M[i][j] = 1;

}

};

Matriz mult(Matriz a,Matriz b){

Matriz ans;

for(int i=0;i<K;i++){

for(int j=0;j<K;j++){

a.M[i][j] %= MOD;

a.M[i][j] += MOD;

a.M[i][j] %= MOD;

b.M[i][j] %= MOD;

b.M[i][j] += MOD;

b.M[i][j] %= MOD;

}

}

for(int i=0;i<K;i++){

for(int j=0;j<K;j++){

for(int k=0;k<K;k++){

ans.M[i][j] += a.M[i][k]\*b.M[k][j];

ans.M[i][j] %= MOD;

ans.M[i][j] += MOD;

ans.M[i][j] %= MOD;

}

}

}

return ans;

}

Matriz pot(Matriz a,ll b){

for(int i=0;i<K;i++){

for(int j=0;j<K;j++){

a.M[i][j] %= MOD;

a.M[i][j] += MOD;

a.M[i][j] %= MOD;

}

}

Matriz ans;ans.iden();

if(b==0) return ans;

if(b==1) return a;

if(b%2==1) ans = a;

Matriz val = pot(a,b/2);

ans = mult(ans,val);

ans = mult(ans,val);

return ans;

}

int main(){

//a(n) = a(n-1) + 5a(n-2) + a(n-3) - a(n-4)

/\*

|0 0 0 -1| |1 | |

|1 0 0 1 | \* |5 | = |

|0 1 0 5 | |11| |

|0 0 1 1 | |36| |

\*/

ll n;cin>>n;

if(n==1) cout<<"1\n";

else if(n==2) cout<<"5\n";

else if(n==3) cout<<"11\n";

else if(n==4) cout<<"36\n";

else{

Matriz ans;

ans.M[0][0] = 0LL;

ans.M[0][1] = 0LL;

ans.M[0][2] = 0LL;

ans.M[0][3] = (MOD-1)\*1LL;

ans.M[1][0] = 1LL;

ans.M[1][1] = 0LL;

ans.M[1][2] = 0LL;

ans.M[1][3] = 1LL;

ans.M[2][0] = 0LL;

ans.M[2][1] = 1LL;

ans.M[2][2] = 0LL;

ans.M[2][3] = 5LL;

ans.M[3][0] = 0LL;

ans.M[3][1] = 0LL;

ans.M[3][2] = 1LL;

ans.M[3][3] = 1LL;

ans = pot(ans,n-4);

vector<ll> a(4);

a[0] = 1LL;

a[1] = 5LL;

a[2] = 11LL;

a[3] = 36LL;

ll res = 0;

for(int i=0;i<4;i++){

res += a[i]\*ans.M[i][3];

res %= MOD;

res += MOD;

res %= MOD;

}

cout<<res<<endl;

}

}

**HARD EQUATION (A^X = B MOD M)**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

ll a,b,m;

ll pot(ll x,ll y,ll mod){

if(y==0) return 1LL;

if(y==1) return x;

ll ans = 1;

if(y&1) ans = x;

ll val = pot(x,y/2,mod);

ans \*= val;

ans %= mod;

ans \*= val;

ans %= mod;

return ans;

}

void solve(){

cin>>a>>b>>m;

a%=m;

b%=m;

if(m==1){

if(a==0) cout<<1<<'\n';

else cout<<0<<'\n';

return;

}

if(b==1){

cout<<0<<'\n';

return;

}

ll n = 1;

while(n\*n<m) n++;

map<ll,int> M;

ll base = pot(a,n,m);

ll curPot = base;

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

M[curPot] = i;

curPot \*= base;

curPot %= m;

}

ll vali = 1;

for(int i=0;i<n;i++){

ll cur = vali\*b;

cur %= m;

if(M.count(cur)){

cout<<M[cur]\*n-i<<'\n';

return;

}

vali\*=a;

vali%=m;

}

}

int main(){

ios::sync\_with\_stdio(0);

cin.tie(NULL);

int t;cin>>t;

while(t--) solve();

return 0;

}

**PROBABILIDAD DE QUE 2 POSICIONES SE ENCUENTREN CAMBIADAS**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef long double ld;

const ld EPS = (1e-9);

int px,py,n;

ld p;

ld memo[50][50][3001];

bool vis[50][50][3001];

bool vis2[50][3001];

ld dp(int posx,int posy,int k){

if(k==0){

if(posx==py && posy==px) return 1.0L;

else return 0.0L;

}

if(vis[posx][posy][k]) return memo[posx][posy][k];

vis[posx][posy][k] = 1;

ld &ans = memo[posx][posy][k] = 0.0L;

ans = (1.0L-p)\*dp(posx,posy,k-1);

int resta = n-1;

if(abs(posx-posy)==1){

ans += (p/(ld)(n-1))\*dp(posy,posx,k-1);

resta--;

}

if(posx>0 && posx-1!=posy) ans += (p/(ld)(n-1))\*dp(posx-1,posy,k-1),resta--;

if(posy>0 && posy-1!=posx) ans += (p/(ld)(n-1))\*dp(posx,posy-1,k-1),resta--;

if(posx<n-1 && posx+1!=posy) ans += (p/(ld)(n-1))\*dp(posx+1,posy,k-1),resta--;

if(posy<n-1 && posy+1!=posx) ans += (p/(ld)(n-1))\*dp(posx,posy+1,k-1),resta--;

if(resta>0) ans += ((ld)resta\*p/(ld)(n-1))\*dp(posx,posy,k-1);

return ans;

}

ld memo2[50][3001];

ld dp2(int posx,int k){

if(k==0){

if(posx==px) return 1.0L;

else return 0.0L;

}

if(vis2[posx][k]) return memo2[posx][k];

vis2[posx][k] = 1;

ld &ans = memo2[posx][k] = 0.0;

ans += (1.0L-p)\*dp2(posx,k-1);

int resta = n-1;

if(posx>0) ans += (p/(ld)(n-1))\*dp2(posx-1,k-1),resta--;

if(posx<n-1) ans += (p/(ld)(n-1))\*dp2(posx+1,k-1),resta--;

if(resta>0) ans += ((ld)resta\*p/(ld)(n-1))\*dp2(posx,k-1);

return ans;

}

int caso = 1;

void solve(){

int k;

memset(vis,0,sizeof vis);

memset(vis2,0,sizeof vis2);

int x,y;

cin>>n>>p>>x>>y>>k;

if(n==1){

printf("Case %d: %.5f\n",caso++,1.0);

return;

}

if(p<EPS){

if(x==y) printf("Case %d: %.5f\n",caso++,1.0);

else printf("Case %d: %.5f\n",caso++,0.0);

return;

}

px = x;

py = y;

if(x!=y) printf("Case %d: %.5f\n",caso++,(double)dp(x,y,k));

else printf("Case %d: %.5f\n",caso++,(double)dp2(x,k));

}

int main(){

//freopen ("assessment.in","r",stdin);

int t;cin>>t;

while(t--) solve();

return 0;

}

**MINIMO RADIO QUE INCLUYE TODOS LOS PUNTOS Y ES TANGENTE A RECTA**

#include<bits/stdc++.h>

using namespace std;

#define mp make\_pair

typedef long long ll;

typedef long double ld;

const int N = (1e5);

const ld INF = (1e16L);

const ld EPS = (1e-9);

struct Point{

ld x,y;

Point(){}

pair<ld,ld> getRadio(ld r){

ld b = 2\*x;

ld c = x\*x+y\*y-2\*y\*r;

ld d = b\*b-4\*c;

if(d<EPS) return mp(1.0L,0.0L);

d = sqrt(d);

return mp(x-d/2.0L,x+d/2.0L);

}

} P[N+2];

int n;

bool f(ld x){

ld left = -INF,right = INF;

for(int i=0;i<n;i++){

pair<ld,ld> cur = P[i].getRadio(x);

left = max(left,cur.first);

right = min(right,cur.second);

}

if(right+EPS <= left) return false;

return true;

}

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

cin>>n;

for(int i=0;i<n;i++) cin>>P[i].x>>P[i].y;

int ok=0;

for(int i=0;i<n;i++) ok|=(P[i].y<0?1:2);

if(ok==3){

cout<<-1<<'\n';

return 0;

}

for(int i=0;i<n;i++) P[i].y=abs(P[i].y);

ld lo=0.0L,hi=INF;

for(int i=0;i<100;i++){

ld mi = (hi+lo)/2.0L;

if(f(mi))hi=mi;

else lo=mi;

}

printf("%.10f\n",(double)hi);

return 0;

}

**PUNTO QUE INCLUYE MAYOR CANTIDAD DE CIRCULOS**

**//SE UTILIZA INTERSECCION DE CIRCULOS**

#include<bits/stdc++.h>

using namespace std;

#define Vector Point

#define pb push\_back

typedef long long ll;

typedef long double ld;

const ld EPS = (1e-9);

ld sqr(ld x){

return x\*x;

}

struct Point{

ld x,y;

Point(){}

Point(ld \_x,ld \_y){

x = \_x;

y = \_y;

}

ld mod(){return sqrt(sqr(x)+sqr(y));}

Point ort(){return Point(-y,x);}

Point unit(){

ld k = mod();return Point(x/k,y/k);

}

};

Point operator +(const Point &a,const Point &b){

return Point(a.x+b.x,a.y+b.y);

}

Point operator -(const Point &a,const Point &b){

return Point(a.x-b.x,a.y-b.y);

}

Point operator \*(const Point &a,ld k){

return Point(a.x\*k,a.y\*k);

}

ld dist(Point a,Point b){

return sqrt(sqr(a.x-b.x) + sqr(a.y-b.y));

}

struct Circle{

Point c;

ld r;

Cirle(){}

bool in(Point x){

ld d = dist(x,c);

return (d<=r+EPS);

}

void show(){

c.show();

cout<<r<<endl;

}

};

vector<Point> circleCircleIntersection(Circle x,Circle y){

vector<Point> ans;

ld d = dist(x.c,y.c);

if(d<EPS) return ans;

if(d>x.r+y.r || d<abs(x.r-y.r)) return ans;

else{

ld a = (sqr(x.r)-sqr(y.r)+d\*d)/(2.0\*d);

ld b = d-a;

ld c = sqrt(abs(sqr(x.r)-sqr(a)));

Vector V = (y.c-x.c).unit();

Point H = x.c + V\*a;

ans.pb(H+V.ort()\*c);

if(c>EPS) ans.pb(H-V.ort()\*c);

return ans;

}

}

bool cmp(pair<Circle,ll> x,pair<Circle,ll> y){

return x.second>y.second;

}

void solve(){

int n,m;cin>>n>>m;

vector< pair<Circle,ll> > v(n);

for(int i=0;i<n;i++) cin>>v[i].first.c.x>>v[i].first.c.y>>v[i].first.r>>v[i].second;

sort(v.begin(),v.end(),cmp);

vector<Point> inter;

for(int i=0;i<n;i++){

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

vector<Point> cur = circleCircleIntersection(v[i].first,v[j].first);

for(int k=0;k<cur.size();k++) inter.pb(cur[k]);

}

inter.pb(v[i].first.c);

}

ll ans = 0;

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

ll val = 0,k=0;

for(int j=0;j<n&&k<m;j++){

if(v[j].first.in(inter[i])){

val+=v[j].second;

k++;

}

}

ans =max(ans,val);

}

cout<<ans<<'\n';

}

int main(){

//ios::sync\_with\_stdio(0);cin.tie(NULL);cout.tie(NULL);

int t;cin>>t;

while(t--) solve();

return 0;

}

**ENCONTRAR 2 CIRCULOS CONCENTRICOS QUE CUBREN TODOS LOS PUNTOS**

#include <bits/stdc++.h>

using namespace std;

typedef long double ld;

const ld DINF = (1e100);

const ld EPS = (1e-9);

struct Point{

ld x,y;

Point(){}

Point(ld \_x,ld \_y){

x = \_x;

y = \_y;

}

ld norm2(){

return \*this\*\*this;

}

ld norm(){

return sqrt(norm2());

}

bool operator ==(Point p){

return abs(x-p.x)<EPS && abs(y-p.y)<EPS;

}

Point operator +(Point p){

return Point(x+p.x,y+p.y);

}

Point operator -(Point p){

return Point(x-p.x,y-p.y);

}

Point operator \*(ld t){

return Point(x\*t,y\*t);

}

Point operator /(ld t){

return Point(x/t,y/t);

}

ld operator \*(Point p){

return x\*p.x+y\*p.y;

}

Point unit(){

return \*this/norm();

}

ld operator %(Point p){

return x\*p.y-y\*p.x;

}

Point ort(){

return Point(-y,x);

}

bool operator<(Point p)const{

return (x<p.x-EPS)||(abs(x-p.x)<EPS&&y<p.y-EPS);

}

bool left(Point p,Point q){

return (q-p)%(\*this-p)>-EPS;

}

};

struct linea{

Point p,pq;

linea(){}

linea(Point p,Point q):p(p),pq(q-p){}

bool operator/(linea l){

return abs(pq.unit()%l.pq.unit())<EPS;

}

Point operator ^(linea l){

if(\*this/l) return Point(DINF,DINF);

Point r = l.p + l.pq\*((p-l.p)%pq/(l.pq%pq));

return r;

}

};

vector<Point> v;

struct radio{

ld r;

radio(ld \_r){

r = \_r;

}

bool operator ==(radio p){

return abs(r-p.r)<EPS;

}

};

bool operator <(radio a,radio b){

return a.r<b.r-EPS;

}

linea mediatriz(Point a,Point b){

Point mid = (a+b)/2.0L;

Point vec = (a-b).ort();

return linea(mid,mid+vec);

}

Point centro(Point a,Point b,Point c){

return mediatriz(a,b)^mediatriz(c,b);

}

int n;

bool areLinea(){

bool ok=1;

for(int i=2;i<n;i++){

if(abs((v[i]-v[i-2])%(v[i-1]-v[i-2])) >= EPS) ok=0;

}

return ok;

}

int main() {

cin>>n;

v.resize(n);

for(int i=0;i<n;i++) cin>>v[i].x>>v[i].y;

if(n==2){

cout<<"INF\n";

return 0;

}

ld ans = 0.0L;

if(n==4){//no es linea

if(areLinea()){

vector<int> idx(4);

for(int i=0;i<4;i++) idx[i]=i;

do{

Point center = (v[idx[0]]+v[idx[1]])/2.0L;

set<radio> S;

for(int m=0;m<n;m++){

S.insert(radio((v[m]-center).norm()));

}

if(S.size()>2){

continue;

}

if(S.size()==2){

ld radio1 = (\*S.begin()).r;

ld radio2 = (\*S.rbegin()).r;

ans = max(ans,(radio2-radio1)/2.0L);

}else{

cout<<"INF\n";

return 0;

}

}while(next\_permutation(idx.begin(),idx.end()));

}else{

vector<int> idx(4);

for(int i=0;i<4;i++) idx[i]=i;

do{

Point center = mediatriz(v[idx[0]],v[idx[1]])^mediatriz(v[idx[2]],v[idx[3]]);

if(center == Point(DINF,DINF)) continue;

set<radio> S;

for(int m=0;m<n;m++){

S.insert(radio((v[m]-center).norm()));

}

if(S.size()>2){

continue;

}

if(S.size()==2){

ld radio1 = (\*S.begin()).r;

ld radio2 = (\*S.rbegin()).r;

ans = max(ans,(radio2-radio1)/2.0L);

}else{

cout<<"INF\n";

return 0;

}

}while(next\_permutation(idx.begin(),idx.end()));

sort(idx.begin(),idx.end());

do{

Point center = centro(v[idx[0]],v[idx[1]],v[idx[2]]);

if(center == Point(DINF,DINF)) continue;

set<radio> S;

for(int m=0;m<n;m++){

S.insert(radio((v[m]-center).norm()));

}

if(S.size()>2){

continue;

}

if(S.size()==2){

ld radio1 = (\*S.begin()).r;

ld radio2 = (\*S.rbegin()).r;

ans = max(ans,(radio2-radio1)/2.0L);

}else{

cout<<"INF\n";

return 0;

}

}while(next\_permutation(idx.begin(),idx.end()));

}

}

if(n==3){

if(areLinea()){

vector<int> idx(3);

for(int i=0;i<3;i++) idx[i]=i;

do{

Point center = (v[idx[0]]+v[idx[1]])/2.0L;

set<radio> S;

for(int m=0;m<n;m++){

S.insert(radio((v[m]-center).norm()));

}

if(S.size()>2){

continue;

}

if(S.size()==2){

ld radio1 = (\*S.begin()).r;

ld radio2 = (\*S.rbegin()).r;

ans = max(ans,(radio2-radio1)/2.0L);

}else{

cout<<"INF\n";

return 0;

}

}while(next\_permutation(idx.begin(),idx.end()));

}

else{

cout<<"INF\n";

return 0;

}

}

if(n>4){

vector<int> idx(5);

for(int i=0;i<5;i++) idx[i]=i;

do{

Point cent = centro(v[idx[0]],v[idx[1]],v[idx[2]]);

if(cent==Point(DINF,DINF)) continue;

set<radio> S;

for(int m=0;m<n;m++){

S.insert(radio((v[m]-cent).norm()));

}

if(S.size()>2){

continue;

}

if(S.size()==2){

ld radio1 = (\*S.begin()).r;

ld radio2 = (\*S.rbegin()).r;

ans = max(ans,(radio2-radio1)/2.0L);

}else{

cout<<"INF\n";

return 0;

}

}while(next\_permutation(idx.begin(),idx.end()));

}

if(ans<EPS){

cout<<"NO\n";

}else printf("%.2f\n",(double)ans);

return 0;

}

**CUANTOS PARES DE PUNTOS POSEEN IGUAL X o Y**

#include<bits/stdc++.h>

#define mp make\_pair

using namespace std;

int a [ 100002 ] ;

typedef long long ll;

map<int,int> X,Y;

map<pair<int,int>,int> XY;

ll f ( ll x ) { return ( x \* x - x ) / 2LL ; }

int main(){

ios::sync\_with\_stdio(0);

cin.tie(NULL);

cout.tie(NULL);

int n ;

cin >> n ;

for ( int i = 0 ; i < n ; i ++ ) {

int x , y ;

cin >> x >> y ;

if ( !X.count(x) ) X [ x ] = 0 ;

X [ x ] ++ ;

if ( !Y.count(y) ) Y [ y ] = 0 ;

Y [ y ] ++ ;

if ( !XY.count(mp(x,y)) ) XY [ mp(x,y) ] = 0 ;

XY [ mp(x,y) ] ++ ;

}

ll ans = 0 ;

for ( auto u : X ) ans += f ( u.second ) ;

for ( auto u : Y ) ans += f ( u.second ) ;

for ( auto u : XY ) ans -= f ( u.second ) ;

cout << ans << endl ;

}

**INVERSA DE MATRIX USANDO BITSET (O(N/32)^3)**

#include<bits/stdc++.h>

using namespace std;

const int N = (2000);

int D[N+2][N+2], nodo[N+2][N+2];

bitset<N> M[2\*N + 2];

int n;

int main(){

memset(nodo,-1,sizeof nodo);

int m;cin>>n>>m;

int a,b;

for(int i=0;i<m;i++){

cin>>a>>b;

a--;b--;

M[a].set(n-b-1);

nodo[a][b] = i;

}

for(int i=n;i<2\*n;i++){

M[i].set(2\*n-i-1);

}

//sacando inversa

bool ok=0;

for(int j=0;j<n;j++){

if(!M[j].test(n-j-1)){//esta apagado el bit que pertenece a la identidad

int change = -1;

for(int k=j+1;k<n;k++){

if(M[k].test(n-j-1)){

change = k;

break;

}

}

swap(M[j],M[change]);

swap(M[j+n],M[change+n]);

}

for(int i=0;i<n;i++){

if(i==j) continue;

if(M[i].test(n-j-1)){

//flipamos

M[i]^=M[j];

M[i+n]^=M[j+n];

}

}

}

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

if(M[i+n].test(n-j-1)) D[j][i] = 1;

else D[j][i] = 0;

}

}

vector<string> ans(m);

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

if(nodo[i][j]==-1) continue;

if(D[i][j]) ans[nodo[i][j]] = "NO";

else ans[nodo[i][j]] = "YES";

}

}

for(int i=0;i<m;i++) cout<<ans[i]<<'\n';

return 0;

}

**MATRIX 2^N\*2^N, CON UN ESPACIO EN BLANCO,LLENADO POR DOMINO L**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = (1<<9);

ll M[ N+2 ][ N+2 ];

void fill(int len,int x,int y,int xx,int yy,ll &val){//xx,yy punto inicial, x,y

if(len==0) return;

len--;

int pot = (1<<len);

pair<int,int> a[] = {{xx,yy},{xx+pot,yy},{xx,yy+pot},{xx+pot,yy+pot}};

pair<int,int> f[] = {{xx+pot-1,yy+pot-1},{xx+pot+pot-1,yy+pot-1},{xx+pot-1,yy+pot+pot-1},{xx+pot+pot-1,yy+pot+pot-1}};

pair<int,int> b[] = {{xx+pot-1,yy+pot-1},{xx+pot,yy+pot-1},{xx+pot-1,yy+pot},{xx+pot,yy+pot}};

int index=val;

for(int i=0;i<4;i++){

if(x>=a[i].first && y>=a[i].second && x<=f[i].first && y<=f[i].second){

if(len>0) fill(len,x,y,a[i].first,a[i].second,++val);

}else{

M[b[i].first][b[i].second]=index;

if(len>0) fill(len,b[i].first,b[i].second,a[i].first,a[i].second,++val);

}

}

}

int main(){

int n,x,y;cin>>n;

cin>>x>>y;

ll val=1;

fill(n,x,y,1,1,val);

int len = (1<<n);

for(int i=1;i<=len;i++)for(int j=1;j<=len;j++) cout<<M[i][j]<<(char)(j==len?10:32);

}

**BIT QUERY Y UPDATE EN RANGE**

#include <bits/stdc++.h>

using namespace std;

#define fast\_io() ios\_base::sync\_with\_stdio(0);cin.tie(0)

#define fi first

#define se second

#define endl '\n'

typedef long long ll ;

// BIT: Query y Update en un Rango [L,R]

const ll MAXN = 100008 ;

struct ft{ // Indexado de 1

ll tree1[ MAXN+8 ] , tree2[ MAXN+8 ] ;

ft(){}

void init(int m , ll val = 0 ){

for(int i=0;i<=m;i++) tree1[i]=val, tree2[i]=val;

}

ll query1( ll i , ll num = 0 ){

while(i>0) num+= tree1[i], i-=(i&-i);

return num ;

}

void update1( ll i, ll del){

while( i<=MAXN )tree1[i]+=del, i+=(i&-i);

return;

}

ll query2( ll i , ll num = 0 ){

while(i>0) num+= tree2[i], i-=(i&-i);

return num ;

}

void update2( ll i, ll del){

while( i<=MAXN )tree2[i]+=del, i+=(i&-i);

return;

}

void update( ll l , ll r , ll val ){ // [ l , r ] acotados

update1(l,val); update1(r+1,-val);

update2(l,val\*(l-1)); update2(r+1,-val\*r);

return;

}

ll query( ll l , ll r ){ // [ l , r ] acotados

ll a= query1(r)\*r-query2(r) , b= query1(l-1)\*(l-1)-query2(l-1);

return a-b ;

}

};

ft ft1,ft2 ;

ll valh[MAXN+5],valm[MAXN+5];

int main(){

fast\_io();

int n;cin>>n;

ft1.init(n);

ft2.init(n);

string s;cin>>s;

for(int i=0;i<n-1;i++){

if(s[i]=='H')ft1.update(1,n-i-1,1);

else ft2.update(1,n-i-1,1);

}

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

valh[i]=ft1.query(i,i);

valm[i]=ft2.query(i,i);

}

for(int i=n-1;i>0;i--){

if(valh[i]>1){

valh[i-1]+=valh[i]/2;

valh[i]%=2;

}

if(valm[i]>1){

valm[i-1]+=valm[i]/2;

valm[i]%=2;

}

}

for(int i=0;i<=n;i++){

if(valh[i]>valm[i]){

cout<<'H'<<endl;

return 0;

}

if(valh[i]<valm[i]){

cout<<'M'<<endl;

return 0;

}

}

cout<<"HM"<<endl;

return 0 ;

}

**SEGMENT TREE LAZY PROPAGATION**

#include<bits/stdc++.h>

#define fi first

#define se second

using namespace std;

typedef long long ll;

const int MAXN = 500002 ;

struct T{

int cnt,mi;

T(){ cnt=mi=0;}

T(int \_cnt,int \_mi ) { cnt = \_cnt , mi = \_mi ; }

void add ( int x ) { mi += x ; }

} tree [ MAXN \* 4 ] ;

T operator + ( T l , T r ) {

if ( l.mi == r.mi ) return T ( l.cnt + r.cnt , l.mi ) ;

return l.mi < r.mi ? l : r ;

}

int lz [ MAXN \* 4 ] ;

void build ( int node , int a , int b ) {

if ( a == b ) {

tree [ node ] = T ( 1 , 0 ) ;

return ;

}

int mid=(a+b)>>1;

build ( node<<1, a,mid);

build ( node<<1|1, mid+1,b);

tree[node] = tree[node<<1] + tree[node<<1|1] ;

}

void update ( int node, int a, int b, int i, int j, int value) {

if( lz[node] ) {

tree [ node ].add ( lz [ node ] ) ;

if ( a != b ) {

lz [ node << 1 ] += lz [ node ] ;

lz [ node << 1 | 1 ] += lz [ node ] ;

}

lz [ node ] = 0 ;

}

if ( a > b || a > j || b < i) return;

if ( a >= i && b <= j) {

tree [ node ].add ( value ) ;

if ( a != b ) {

lz [ node << 1 ] += value ;

lz [ node << 1 | 1 ] += value ;

}

return;

}

int mid=(a+b)>>1;

update(node<<1, a,mid, i, j, value);

update(node<<1|1, mid+1, b, i, j, value);

tree[node] = tree[node<<1] + tree[node<<1|1];

}

T query(int node, int a, int b, int i, int j) {

if ( a > b || a > j || b < i ) return T() ;

if( lz[node] ) {

tree [ node ].add ( lz [ node ] ) ;

if ( a != b ) {

lz [ node << 1 ] += lz [ node ] ;

lz [ node << 1 | 1 ] += lz [ node ] ;

}

lz [ node ] = 0 ;

}

if (a >= i && b <= j ) return tree [ node ] ;

int mid=(a+b)>>1;

return query ( node << 1 , a , mid , i , j ) + query ( node << 1 | 1 , mid + 1 , b , i , j ) ;

}

int a [ MAXN ] , p [ MAXN ] ;

map<int,pair<int,int> > MAPA ;

map<int,int> PERSONA ;

pair<int,int> Q [ MAXN ] ;

int solve ( int n ){

T ans = query ( 1 , 0 , n - 1 , 0 , n - 1 ) ;

if ( ans.mi != 0 ) return n ;

return n- ans.cnt ;

}

int main() {

int n , m , d , l ;

cin >> n >> m >> d >> l ;

build ( 1 , 0 , n -1 ) ;

for ( int i = 1 ; i < n ; i ++ ) cin >> a [ i ] ;

set<int> s;

for ( int i = 0 ; i < m ; i ++ ) cin >> p [ i ] , s.insert ( p [ i ] ) ;

for ( int i = 0 ; i < d ; i ++ ) {

cin >> Q [ i ].fi >> Q [ i ].se ;

s.insert ( Q [ i ].se ) ;

}

for ( auto u : s ) {

int lo = lower\_bound(a,a+n,u-l) - a;

int hi = lower\_bound(a,a+n,u+l+1) - a;

MAPA[u]=make\_pair(lo,hi-1);

}

s.clear();

for ( int i = 0 ; i < m ; i ++ ) {

PERSONA[p[i]]= i;

pair<int,int> z = MAPA [ p [ i ] ] ;

if(z.fi<=z.se)update ( 1 , 0 , n - 1 , z.fi , z.se , 1 ) ;

}

cout<< solve( n ) << endl ;

for ( int i = 0 ; i < d ; i ++ ) {

int u = Q [ i ].fi , y = Q [ i ].se ;

int x = PERSONA [ u ] ;

pair<int,int> z = MAPA [ u ] ;

if(z.fi<=z.se)update ( 1 , 0 , n - 1 , z.fi , z.se , -1 ) ;

PERSONA.erase( u ) ;

u = y ;

PERSONA [ u ] = x ;

z = MAPA [ u ] ;

if(z.fi<=z.se)update ( 1 , 0 , n - 1 , z.fi , z.se , 1 ) ;

cout << solve ( n ) << endl ;

}

}

**USO DE LA LIBERIA JSON PYTHON3**

import json

def getval(data):

for i in range(len(data)):

if(data[i]<i+1):

return i

return len(data)

n = int(input())

thisdistc = {}

for i in range(n):

s = input()

thatjson = json.loads(s)

autores = thatjson["authors"]["authors"]

cntcitas = int(thatjson["citing\_paper\_count"])

for x in autores:

name = x["full\_name"]

if name in thisdistc:

thisdistc[name].append(cntcitas)

else:

thisdistc[name] = []

thisdistc[name].append(cntcitas)

for x in thisdistc:

thisdistc[x].sort(reverse=True)

dictans = {}

for x in thisdistc:

value = getval(thisdistc[x])

if value in dictans:

dictans[value].append(x)

else:

dictans[value]=[]

dictans[value].append(x)

for x in dictans:

dictans[x].sort()

for i in reversed(range(1000)):

if i in dictans:

for x in dictans[i]:

print(x,i)