```cpp
#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast,unroll-loops")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define dbg(x) cerr << #x << " = " << x << endl;
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef long double ld;
typedef pair<int,int> ii;

const int MOD = 998244353;
const int MAXN = 5e5+2;

int fact[MAXN], inv[MAXN], factInv[MAXN];

void precomp() {
    fact[0] = 1;
    forsn(i,1,MAXN) fact[i] = (fact[i-1]*(ll)i)%MOD;

    inv[1] = 1;
    forsn(i,2,MAXN) inv[i] = MOD - ((MOD/(ll)i)*(ll)inv[MOD%i])%MOD;

    factInv[0] = 1;
    forsn(i,1,MAXN) factInv[i] = (factInv[i-1]*(ll)inv[i])%MOD;
}

int nCr (int n, int r) {
    if (r > n) return 0;
    return ((ll)((fact[n]*(ll)factInv[r])%MOD)*factInv[n-r])%MOD;
}

int main() {
    FAST_IO;

    precomp();

    int n,k; cin >> n >> k;

    int rta = 0;
    forsn(i,1,n+1) {
        int cantDiv = n/i;
        rta = (rta + nCr(cantDiv-1,k-1))%MOD;
    }

    cout << rta;

    return 0;
}

/// ¡¡¡¡¡ HACE CASOS DE PRUEBAAAAAAAAAAAAAAAA !!!!!!!!!
/// ESCRIBÍ en vez de tanto dar vueltas
/// si te parece que no va PROBALO PRIMERO!
/// CODEA LO BÁSICO PRIMERO!

#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast,unroll-loops")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define dbg(x) cerr << #x << " = " << x << endl;
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);
```

```cpp
using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef long double ld;
typedef pair<int,int> ii;

const int MAXN = 2e5+5;
const int MAXST = (1<<(32-__builtin_clz(MAXN)));
const int INF = 2e9;

void fs (int &x) {
    x = 0;
    int c; c = getchar();
    if (c < '0' || c > '9') c = getchar();
    for(; c >= '0' && c <= '9'; c = getchar())
        x = 10*x + c-'0';
}

int n,k1,k2;
ll rta = 0;

struct nodo {
    int l,r,sum,p;
    nodo() : l(-1), r(-1), sum(0), p(-1) {};
};

struct SegT {
    vector<nodo> bag;

    SegT() : bag({nodo()}) {};

    int newNode (int i) { // bolsa de nodos, pero no es persistent segtree
        if (i == -1) {bag.pb(nodo()); return (int)bag.size()-1;}
        return i;
    }

    int update (int i, int p, int v, int tl = 0, int tr = MAXST) {
        if (tl > p || tr <= p) return i;
        i = newNode(i);
        if (tr-tl == 1) { // cuando lo creo, debe saber quién es
            bag[i].sum += v, bag[i].p = p;
        }
        else {
            int mid = (tl+tr)/2;
            bag[i].l = update(bag[i].l,p,v,tl,mid);
            bag[i].r = update(bag[i].r,p,v,mid,tr);
            bag[i].sum = (bag[i].l == -1 ? 0 : bag[bag[i].l].sum) + (bag[i].r == -1 ? 0 : bag[bag[i].r].sum);
        }

        return i;
    }

    int query (int i, int bl, int br, int tl = 0, int tr = MAXST) {
        if (tl >= br || tr <= bl || i == -1) return 0;
        if (tl >= bl && tr <= br) return bag[i].sum;

        int mid = (tl+tr)/2;
        return query(bag[i].l,bl,br,tl,mid) + query(bag[i].r,bl,br,mid,tr);
    }
};

struct dato {
    SegT ST;
    int minDep;
};

struct DS {
    vi p,r;
    vector<dato> cmp;

    void init(int N) {
        p.assign(N,0);
        r.assign(N,0);
        cmp.assign(N,{SegT(),INF});
        forn(i,N) p[i] = i;
    }

    int find (int x) {return p[x] == x ? x : p[x] = find(p[x]);}
```

```
    void merge (int y, int x) {
        int m_minDep = min(cmp[x].minDep,cmp[y].minDep);
        vector<ii> toUpd;

        for (auto &i : cmp[x].ST.bag) {
            if (i.p != -1) { // si es una hoja
                int left = k1-i.p+2*m_minDep;
                int right = k2-i.p+2*m_minDep; // se pueden ir a
↪   negativos o positivos, pero como el ST busca en un rango dado, no
↪   hay drama
                int tot = cmp[y].ST.query(0,left,right+1);

                rta += (ll)tot * i.sum;

                toUpd.pb({i.p, i.sum});
            }
        }

        for (auto &i : toUpd) cmp[y].ST.update(0,i.fst,i.snd);
        cmp[y].minDep = m_minDep;
    }

    bool join (int a, int b) {
        int x = find(a), y = find(b);
        if (x == y) return false;
        if (r[x] > r[y]) swap(x,y);
        p[x] = y; merge(y,x);
        if (r[x] == r[y]) r[y]++;
        return true;
    }
};

vi G[MAXN];
bitset<MAXN> done;
DS UF;

void dfs1 (int st, int lvl) {
    done[st] = true;

    UF.cmp[st].minDep = lvl;
    UF.cmp[st].ST.update(0,lvl,1);

    for (auto &i : G[st])
        if (!done[i]) {
            dfs1(i,lvl+1);
            UF.join(st,i);
        }
}

int main() {
    fs(n), fs(k1), fs(k2);

    UF.init(n+2);

    forn(i,n-1) {
        int u,v; fs(u), fs(v); u--, v--;
        G[u].pb(v), G[v].pb(u);
    }

    dfs1(0,0);

    printf("%lld",rta);

    return 0;
}

/// ¡¡¡¡ HACE CASOS DE PRUEBAAAAAAAAAAAAAAAA !!!!!!!!!
/// ESCRIBÍ en vez de tanto dar vueltas
/// si te parece que no va PROBALO PRIMERO!
/// CODEA LO BÁSICO PRIMERO!

#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast,unroll-loops")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define dbg(x) cerr << #x << " = " << x << endl;
```

```cpp
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef long double ld;
typedef pair<int,int> ii;

const int MAXN = 5005;
const ll NEG = -1;

struct stackMax {
    vector< pair<ll,ll> > stck;

    ll maxi() {return (stck.empty() ? NEG : stck.back().snd);}
    pair<ll,ll> top() {return stck.back();}
    void push(ll x) {stck.pb({x, max(x, this->maxi())});}
    void pop() {if (!stck.empty()) stck.pop_back();}
    void clear() {stck.clear();}
    bool empty() {return stck.empty();};
    int size() {return (int)stck.size();}
};

struct queueMax {
    stackMax s1,s2;

    ll maxi() {return max( s1.maxi(), s2.maxi() );}
    void pop() {
        if (s2.empty()) while (!s1.empty())
            s2.push(s1.top().fst), s1.pop();
        s2.pop();
    }
    void push(ll x) {s1.push(x);}
    int size() {return (int)s1.size() + (int)s2.size();}
    void clear() {s1.clear(), s2.clear();}
};
```

```cpp
int arr[MAXN];
ll dp[2][MAXN];

int main() {
    FAST_IO;

    int n,k,x; cin >> n >> k >> x;

    forn(i,n) cin >> arr[i+1];

    if (n/k > x) return cout << -1, 0;

    // build -> nada porque ya es todo 0
    forn(i,MAXN) dp[0][i] = dp[1][i] = -1; // init
    forn(i,k) dp[0][i] = 0; // puntos de partida

    queueMax Q;
    forsn(j,1,x+1) {
        Q.clear();
        forsn(i,1,n+1) {
            ll actVal = dp[1^(j&1)][i-1]; dp[1^(j&1)][i-1] = -1; //
↪   reset mientras de lo que ya no uso
            Q.push(actVal >= 0 ? actVal + arr[i] : -1);
            if (Q.size() > k) Q.pop();
            dp[j&1][i] = max(dp[j&1][i],Q.maxi()); // el max para
↪   evitar irme por el borde
        }
    }

    cout << dp[x&1][n];

    return 0;
}

#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast,unroll-loops")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
```

```cpp
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define dbg(x) cerr << #x << " = " << x << endl;
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef long double ld;
typedef pair<int,int> ii;

const int MAXN = 2e5+5;
const int MAXST = (1<<(32-__builtin_clz(MAXN)));
const int INF = 2e9;
const int NEUT = -INF;

void fs (int &x) {
    x = 0;
    int c; c = getchar();
    if (c < '0' || c > '9') c = getchar();
    for (; c >= '0' && c <= '9'; c = getchar())
        x = 10*x + c-'0';
}

struct mon {
    int v;

    mon operator+ (const mon &o) const {
        return {max(v,o.v)};
    }
};

vi G[MAXN];
int vals[MAXN], P[MAXN], hd[MAXN], pos[MAXN], hv[MAXN], dep[MAXN],
↪  cntPos = 0;
int n,q,N;
mon ST[2*MAXST];

// HLD (Heavy/Light Decompositiion)
int dfs (int st, int lvl) {
    dep[st] = lvl;

    int maxi = NEUT, ind = -1, cnt = 0;
    for (auto &i : G[st])
        if (P[st] != i) {
            P[i] = st;
            int aux = dfs(i,lvl+1);
            cnt += aux;
            if (aux > maxi) maxi = aux, ind = i;
        }

    hv[st] = ind;
    return cnt+1;
}

void decompose(int st, int m_hd) {
    pos[st] = cntPos++, hd[st] = m_hd;
    if (hv[st] != -1)
        decompose(hv[st],m_hd);
    for (auto &i : G[st])
        if (hv[st] != i && P[st] != i)
            decompose(i,i);
}

void update (int p, int v) {
    p += N; ST[p].v = v;
    while (p > 1)
        p /= 2, ST[p] = ST[2*p] + ST[2*p+1];
}

mon query (int l, int r) {
    mon acc = {NEUT};
    l += N, r += N;
    for (; l < r; l /= 2, r /= 2) {
        if (l&1) acc = (acc + ST[l++]);
        if (r&1) acc = (acc + ST[--r]);
    }
    return acc;
```

```
}

int query_lca (int a, int b) {
    int maxi = NEUT;
    // notar que la condición siempre depende que los caminos heavy no
↪   convergen
    for (; hd[a] != hd[b]; b = P[hd[b]]) {
        if (dep[hd[a]] > dep[hd[b]]) swap(a,b);
        maxi = max(maxi, query(pos[hd[b]],pos[b]+1).v);
    }
    if (pos[a] > pos[b]) swap(a,b);
    maxi = max(maxi, query(pos[a],pos[b]+1).v);
    return maxi;
}

int main() {
    //freopen("test_input.txt","r",stdin);
    fs(n), fs(q);
    forn(i,n) fs(vals[i]);
    forn(i,n-1) {
        int u,v; fs(u), fs(v); u--, v--;
        G[u].pb(v), G[v].pb(u);
    }

    P[0] = -1;
    dfs(0,0); decompose(0,0);

    N = (1<<(32-__builtin_clz(n)));
    //cerr << N << endl;
    forn(i,n) ST[pos[i]+N].v = vals[i];
    dforsn(i,1,N) ST[i] = ST[2*i]+ST[2*i+1];

    forn(i,q) {
        int typ; fs(typ);

        if (typ == 1) {
            int s,x; fs(s), fs(x); s--;
            update(pos[s],x);
        }
        else {
            int a,b; fs(a), fs(b); a--, b--;
            printf("%d ",query_lca(a,b));
        }
    }

    return 0;
}

/// ¡¡¡¡ HACE CASOS DE PRUEBAAAAAAAAAAAAAAAA !!!!!!!!
/// ESCRIBÍ en vez de tanto dar vueltas
/// si te parece que no va PROBALO PRIMERO!
/// CODEA LO BÁSICO PRIMERO!



//
↪  https://www.hackerrank.com/contests/simulacro-2-oia-2020/challenges/problem-2-roadbloc
#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef pair<int,int> ii;

const int MAXN = 102;
const int INF = 2e9+5;

ii P[MAXN];
int D[MAXN];
```

```cpp
int roadblock(int N, int M, vector <int> a, vector <int> b, vector
↪  <int> c) {
    forn(i,MAXN) {D[i] = INF; P[i] = {-1,-1};}
    forn(i,M) {a[i]--; b[i]--;}
    D[0] = 0;
    forn(i,N) forn(j,M) { // Bellman-Ford
        int newD = D[a[j]]+c[j];
        if (newD < D[b[j]]) {D[b[j]] = newD; P[b[j]] = {a[j],j};}
        int newD2 = D[b[j]]+c[j];
        if (newD2 < D[a[j]]) {D[a[j]] = newD2; P[a[j]] = {b[j],j};}
    }
    int bst = D[N-1];
    vi ind;
    for (int i = N-1; P[i].snd != -1; i = P[i].fst)
        ind.pb(P[i].snd);

    int delta = 0;
    for (auto &i : ind) {
        c[i] <<= 1; // duplico
        forn(j,MAXN) D[j] = INF; D[0] = 0;
        forn(j,N) forn(k,M) {
            int newD = D[a[k]]+c[k];
            if (newD < D[b[k]]) D[b[k]] = newD;
            int newD2 = D[b[k]]+c[k];
            if (newD2 < D[a[k]]) D[a[k]] = newD2;
        }
        delta = max(delta,D[N-1]-bst);
        c[i] >>= 1;
    }

    return delta;
}


int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    int N, M;
    cin >> N >> M;
    vector <int> a(M), b(M), c(M);
    for(int i=0; i<M; i++)
        cin >> a[i] >> b[i] >> c[i];

    cout << roadblock(N, M, a, b, c) << endl;

    return 0;
}
```

```cpp
#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast,unroll-loops")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define dbg(x) cerr << #x << " = " << x << endl;
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef long double ld;
typedef pair<int,int> ii;

const int MAXN = 1e6+2;
const int MAXA = 105;
const int INF = 1e9+5;

int D[MAXA][MAXA];
int path[MAXN],P[MAXN];
multiset<pair<ii,int>> prv; // nodo, longitud subseq, último índice

int main() {
    FAST_IO;
```

```
    int n; cin >> n;
    forn(i,n) forn(j,n) {char c; cin >> c; D[i][j] = c-'0';}
    int m; cin >> m;
    forn(i,m) cin >> path[i], path[i]--;

    forn(i,n) forn(j,n) if (!D[i][j]) D[i][j] = INF; // no son
↪   adyacentes, no hay camino aún
    forn(i,n) D[i][i] = 0; // a mi mismo es 0

    // Floyd-Warshall
    forn(k,n) forn(i,n) forn(j,n)
        D[i][j] = min(D[i][j],D[i][k]+D[k][j]);

    prv.insert({{path[0],0},0}); P[0] = -1; // no tiene padre
    forsn(i,1,m) {
        auto it = prv.lower_bound({{path[i],-INF},-INF});
        if (it != prv.end() && (*it).fst.fst == path[i]) prv.erase(it);
↪   // como volví acá, no hay forma para preservar el de antes,
↪   entonces lo saco
        int len = INF;
        for (auto &j : prv)
            if (j.fst.snd+1 < len && D[j.fst.fst][path[i]] == i-j.snd)
↪   len = j.fst.snd+1, P[i] = j.snd;
        prv.insert({{path[i],len},i});
    }

    vi rta;
    for (int i = m-1; i != -1; i = P[i]) rta.pb(path[i]);

    cout << rta.size() << '\n';
    reverse(all(rta));
    for (auto &i : rta) cout << i+1 << ' ';

    return 0;
}

/// ¡¡¡¡¡ HACE CASOS DE PRUEBAAAAAAAAAAAAAAAA !!!!!!!!!
/// ESCRIBÍ en vez de tanto dar vueltas
/// si te parece que no va PROBALO PRIMERO!
/// CODEA LO BÁSICO PRIMERO!
```

```
#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast,unroll-loops")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define dbg(x) cerr << #x << " = " << x << endl;
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef long double ld;
typedef pair<int,int> ii;

const int P = 31, PP = 53;
const int MOD = 1e9+93;
const int INV_MOD = 935483958;
const int INV_MOD2 = 132075484;
const int MAXN = 2e6+5;

ll hashes1[MAXN], hashes2[MAXN];
map<pair<ll,ll>,vi> hashSet;
vector<string> games;
set<int> foundIds;
vi orden;
int found[MAXN];
int n,k,sz;

ll hashIt (const string &s, int p, int m) {
    ll r = 0, potP = 1;
    for (auto &i : s) {
        r = (r + (i-'a'+1) * potP) % m;
        potP = (potP * p) % m;
```

```
    }
    return r;
}

void hashArray (int i_m, int m, int p, ll arr[], const string &s) {
    string aux;
    forn(i,k) aux.pb(s[i]);
    ll act_hash = hashIt(aux,p,m);

    ll lastPotP = 1;
    forn(i,k-1) lastPotP = (lastPotP * p)%m;

    arr[0] = act_hash;
    for (int i = k; i-k+1 < sz; i++) {
        act_hash = (act_hash - (s[i-k] - 'a' + 1) + m)%m;
        act_hash = (act_hash * i_m)%m;
        act_hash = (act_hash + (s[i] - 'a' + 1) * lastPotP)%m;
        arr[i-k+1] = act_hash;
    }
}

int main() {
    FAST_IO;

    cin >> n >> k;
    string s; cin >> s;
    s += s; // lo duplico por conveniencia
    sz = n*k;

    hashArray(INV_MOD,MOD,P,hashes1,s);
    hashArray(INV_MOD2,MOD,PP,hashes2,s);

    forn(i,sz) hashSet[{hashes1[i],hashes2[i]}].pb(i);

    int g; cin >> g;
    forn(i,g) {
        string x; cin >> x;
        games.pb(x);

        ll mHash = hashIt(x,P,MOD), mHash2 = hashIt(x,PP,MOD);
        pair<ll,ll> auxPair = {mHash,mHash2};
        auto it = hashSet.find(auxPair);
        if (it != hashSet.end()) {
            for (auto &j : (*it).snd)
                found[j] = i+1; // así 0 es que no encontré nada
        }
    }

    forn(i,k) {
        foundIds.clear(), orden.clear();
        bool posib = true;

        for (int j = i; j < sz; j += k) {
            if (!foundIds.count(found[j]) && found[j])
                foundIds.insert(found[j]), orden.pb(found[j]);
            else posib = false;
        }

        if (posib) {
            cout << "YES\n";
            for (auto &j : orden) cout << j << ' ';
            return 0;
        }
    }

    cout << "NO";

    return 0;
}

#include <bits/stdc++.h>

#pragma GCC optimize("Ofast")
#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define all(c) (c).begin(),(c).end()
#define pb push_back
```

```cpp
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef pair<int,int> ii;

const int MAXN = 2e5+5;
const int MAXST = (1<<(32-__builtin_clz(MAXN)));

void fastscan (ll &x) {
    int c; x = 0;
    c=getchar_unlocked();
    if (c<'0'||c>'9') c=getchar_unlocked();
    for(; c>='0' && c<='9'; c=getchar_unlocked())
        x = 10*x + c-'0';
}

ll ST[2*MAXST];
int len[2*MAXST];
pair<ll,ll> lazy[2*MAXST]; // si hay en el primero, es sumar delta. si
↪  hay en el segundo, es establecer a x.
int N;

void build() {
    dforsn(i,1,N) ST[i] = ST[2*i]+ST[2*i+1];
}

void getRangesLen(int i, int tl, int tr) {
    len[i] = tr-tl;

    if (tr-tl <= 1) return;
    int mid = (tl+tr)/2;
    getRangesLen(2*i,tl,mid);
    getRangesLen(2*i+1,mid,tr);
}

void passLazy (int i) {
    bool canPass = ((2*i+1) < (2*MAXST));
    if (lazy[i].snd) {
        ST[i] = len[i]*lazy[i].snd;

        if (canPass) {
            lazy[2*i] = {0,lazy[i].snd};
            lazy[2*i+1] = {0,lazy[i].snd};
        }
    }
    ST[i] += len[i]*lazy[i].fst;
    if (canPass) {
        lazy[2*i].fst += lazy[i].fst;
        lazy[2*i+1].fst += lazy[i].fst;
    }
    lazy[i] = {0,0};
}

void upd(int i, int tl, int tr, int bl, int br, const int v, const bool
↪  mode) { // mode == 0 -> increasase, else set
    passLazy(i);
    if (bl >= br) return;
    if (bl == tl && br == tr) {
        if (!mode) lazy[i].fst += v;
        else lazy[i] = {0,v};
        passLazy(i);
    }
    else {
        int mid = (tl+tr)/2;
        upd(2*i,tl,mid,bl,min(br,mid),v,mode);
        upd(2*i+1,mid,tr,max(bl,mid),br,v,mode);
        ST[i] = ST[2*i] + ST[2*i+1];
    }
}

ll get(int i, int tl, int tr, const int L, const int R) {
    if (tr <= L || tl >= R) return 0;
    passLazy(i);
    int mid = (tl+tr)/2;
    if (tr <= R && tl >= L) return ST[i];
    return get(2*i,tl,mid,L,R)+get(2*i+1,mid,tr,L,R);
}
```

```
int main() {
    ll n,q; fastscan(n); fastscan(q);
    N = (1<<(32-__builtin_clz(n)));
    forn(i,n) fastscan(ST[i+N]);
    build(); getRangesLen(1,0,N);

    forn(i,q) {
        ll typ,a,b,x; fastscan(typ);  fastscan(a); fastscan(b); a--;
        if (typ == 1 || typ == 2) fastscan(x);
        if (typ == 1) upd(1,0,N,a,b,x,0);
        else if (typ == 2) upd(1,0,N,a,b,x,1);
        else printf("%lld\n",get(1,0,N,a,b));
    }

    return 0;
}

/// ESCRIBÍ en vez de tanto dar vueltas
/// si te parece que no va PROBALO PRIMERO!
/// CODEA LO BÁSICO PRIMERO!
/// HACE C-A-S-O-S D-E P-R-U-E-B-A.A.A.A.A!!!

int lca(int u, int v){
  if(depth[v] < depth[u])
    swap(u,v);
  v = ancestro(v, depth[v]-depth[u]);
  if(u==v) return u;
  for(int k = log_maxn;k >= 0; k--){
    if(P[u][k] != P[v][k]){
      u=P[u][k];
      v=P[v][k];
    }
  }
  return P[u][0];
}

#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef pair<int,int> ii;

const int MAXN = 102;

struct pt {
    double x,y;
};

struct edge {
    int a,b;
    double w;

    bool operator< (const edge &o) const {
        return w < o.w;
    }
};

struct DS {
    vi p,r;

    void init(int N) {
        p.assign(N,0);
        r.assign(N,0);
        forn(i,N) p[i] = i;
    }
    int find(int x) {return p[x] == x ? x : p[x] = find(p[x]);}
    bool join(int a, int b) {
        int x = find(a), y = find(b);
```

```
        if (x == y) return false;
        if (r[x] > r[y]) swap(x,y);
        p[x] = y; if (r[y] == r[x]) r[y]++;
        return true;
    }
};

int main() {
    FAST_IO;

    int t; cin >> t;
    forn(i,t) {
        DS MST;
        vector<pt> freckles;
        vector<edge> edges;

        int n; cin >> n; MST.init(n);
        forn(j,n) {
            double x,y; cin >> x >> y;
            freckles.pb({x,y});
        }

        forn(j,n) forsn(k,j+1,n)
↪   edges.pb({j,k,hypot(abs(freckles[j].x-freckles[k].x),abs(freckles[j].y-freckles[k].y))});

        sort(all(edges));

        long double trace = 0;
        for (auto &j : edges) if (MST.join(j.a,j.b)) trace += j.w;

        cout << fixed << setprecision(2) << trace << '\n';
        if (i+1 < t) cout << '\n';
    }

    return 0;
}

/// ESCRIB en vez de tanto dar vueltas
/// si te parece que no va PROBALO PRIMERO!
/// CODEA LO BSICO PRIMERO!
```

```
/// HACE C-A-S-O-S D-E P-R-U-E-B-A.A.A.A.A!!!

#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast,unroll-loops")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define dbg(x) cerr << #x << " = " << x << endl;
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef long double ld;
typedef pair<int,int> ii;

const int MAXN = 305;

int posic[MAXN], inDeg[MAXN];
bool supera[MAXN][MAXN],ady[MAXN][MAXN];
vi G[MAXN];

int main() {
    //freopen("entrada.txt","r",stdin);
    int t; scanf("%d",&t);

    forn(o,t) {
        int n; scanf("%d",&n);

        forn(i,n+2) inDeg[i] = 0, G[i].clear(); // reset
        forn(i,n+2) forn(j,n+2) supera[i][j] = ady[i][j] = 0;

        forn(i,n) scanf("%d",&posic[i]), posic[i]--;
```

```
        forn(i,n) forn(j,i) supera[posic[j]][posic[i]] = true;

        int m; scanf("%d",&m);
        forn(i,m) {
            int u,v; scanf("%d %d",&u,&v); u--, v--;
            if (supera[u][v]) swap(u,v);
            ady[u][v] = true;
            G[u].pb(v); inDeg[v]++;
        }

        forn(i,n) {
            forn(j,i) if (!ady[posic[i]][posic[j]])
↪   G[posic[j]].pb(posic[i]), inDeg[posic[i]]++; // si no hay nada que
↪   diga que lo supero, me supera
            forsn(j,i+1,n) if (!ady[posic[j]][posic[i]])
↪   G[posic[i]].pb(posic[j]), inDeg[posic[j]]++; // si no hay nada que
↪   diga que me supera, lo sigo superando
        }

        vi topSort;
        queue<int> Q;
        forn(i,n) if (!inDeg[i]) Q.push(i);

        while (!Q.empty()) {
            auto e = Q.front(); Q.pop();
            topSort.pb(e);

            for (auto &i : G[e])
                if (!(--inDeg[i])) Q.push(i);
        }

        if ((int)topSort.size() < n) puts("IMPOSSIBLE");
        else {
            for (auto &i : topSort) printf("%d ",i+1);
            putchar('\n');
        }
    }

    return 0;
}
```

```
/// ¡¡¡¡¡ HACE CASOS DE PRUEBAAAAAAAAAAAAAAAA !!!!!!!!!
/// ESCRIBÍ en vez de tanto dar vueltas
/// si te parece que no va PROBALO PRIMERO!
/// CODEA LO BÁSICO PRIMERO!



// AC -> Aizu CGL 4B

#include <bits/stdc++.h>

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef pair<int,int> ii;

const double EPS = 1e-6; // margin of error

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

    double norm2() {return (*this) * (*this);} // se demuestra que
↪   devuelve el cuadrado del mo
    double norm() {return sqrt(norm2());} // entonces este devuelve el
↪   mo
    pt operator+ (const pt &o) const {return pt(x+o.x,y+o.y);}
    pt operator- (const pt &o) const {return pt(x-o.x,y-o.y);}
    pt operator* (const double &t) const {return pt(x*t,y*t);}
    pt operator/ (const double &t) const {return pt(x/t,y/t);}
    double operator* (const pt &o) const {return x*o.x + y*o.y;} // dot
```

```
    double operator% (const pt &o) const {return x*o.y - y*o.x;} //
↪   cross
    bool operator< (const pt &o) const { // comp chull
        return (x < o.x || (x == o.x && y < o.y));
    }
    bool operator== (const pt &o) const {return (abs(o.x-x) < EPS and
↪   abs(o.y-y) < EPS);}
    bool left (const pt &a, const pt &b) {return (b-a)%(*this-a) >
↪   EPS;} // left of directed line ab?
    pt unit(){return (*this/norm());} // devuelve el vector unitario,
↪   al dividir por el mo
};

struct ln {
    pt p,pq; // [lambda]v + p || v is pq and p is...well p
    ln(pt p, pt q) : p(p),pq(q-p){} // start point and direction
    ln(){}

    bool operator/ (ln o) {return (abs(pq.unit()%o.pq.unit()) < EPS);}
↪   // recordar que pq preserva la direccilo trato como vector libre
    pt operator^ (const ln &o) const { // intersection (need to
↪   intersect)
        double div = (pq%o.pq);
        return (o.p+o.pq*((pq%(p-o.p))/div)); // "Agusttrick" :v
    }
};

struct poly {
    int n; vector<pt> p;
    poly(vector<pt> _p){p=_p; n = p.size();}
    poly(){}

    double area() {
        double r = 0;
        forn (i,n) r += p[i]%p[(i+1)%n];
        return abs(r)/2;
    }

    poly cut (pt a, pt b) {
        vector<pt> r;
        forn (i,n) {
            bool k = p[i].left(a,b), l = p[(i+1)%n].left(a,b);
            if (k) r.pb(p[i]);
            ln m(p[i],p[(i+1)%n]);
            if (k != l) r.pb(ln(a,b)^m);
        }
        return poly(r);
    }
};

vector<pt> p;
poly mePol;

int main() {
    FAST_IO;

    //freopen("tmp.txt","w",stdout);

    int n; cin >> n;
    forn (i,n) {
        double a,b; cin >> a >> b;
        p.pb({a,b});
    }
    mePol = poly(p);

    int q; cin >> q;
    forn (i,q) {
        double x1,y1,x2,y2; cin >> x1 >> y1 >> x2 >> y2;
        pt A = {x1,y1}, B = {x2,y2};

        cout << setprecision(13) << (mePol.cut(A,B)).area() << '\n';

    }

    return 0;
}

/// ESCRIBen vez de tanto dar vueltas
/// si te parece que no va PROBALO PRIMERO!
/// CODEA LO BSICO PRIMERO!
/// HACE C-A-S-O-S D-E P-R-U-E-B-A.A.A.A.A!!!
```

```cpp
// AC -> Aizu CGL 4A

#include <bits/stdc++.h>

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef pair<int,int> ii;

const int INF = 1e9+5;

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

    pt operator- (const pt &o) const {return pt(x-o.x, y-o.y);}
    double operator% (const pt &o) const {return x*o.y - y*o.x;} //
↪   cross product
    double operator* (const pt &o) const {return x*o.x + y*o.y;} // dot
↪   product
    bool operator< (const pt &o) const{ // chull sort cmp
        return (x < o.x || (x == o.x && y < o.y));
    }
    bool ccw (const pt &a, const pt &b) {return (b-a)%(*this-a) > 0;}
↪   // counter-clockwise
};

vector<pt> chull (vector<pt> p) {
    vector<pt> r;
    sort(all(p));
    forn (i,p.size()) { // lower hull
        while (r.size() >= 2 and r.back().ccw(r[r.size()-2],p[i]))
↪   r.pop_back();
        r.pb(p[i]);
    }
    r.pop_back(); int k = r.size();
    dforn (i,p.size()) { // upper hull
        while (r.size() >= k+2 and r.back().ccw(r[r.size()-2],p[i]))
↪   r.pop_back();
        r.pb(p[i]);
    }
    r.pop_back();
    return r;
}

int main() {
    FAST_IO;

    vector<pt> pts;
    int n; cin >> n;
    forn (i,n) {
        int x,y; cin >> x >> y;
        pts.pb(pt(x,y));
    }

    vector<pt> rta = chull(pts);
    int ind = -1; double mini = INF;
    forn (i,rta.size()) if (rta[i].y < mini) {mini = rta[i].y; ind =
↪   i;}

    cout << rta.size() << '\n';
    forn (i,rta.size()) {
        cout << rta[ind].x << ' ' << rta[ind].y << '\n';
        ind = (ind+1)%(int)(rta.size());
    }

    return 0;
}

/// ESCRIBen vez de tanto dar vueltas
/// si te parece que no va PROBALO PRIMERO!
```

```cpp
/// CODEA LO BSICO PRIMERO!
/// HACE C-A-S-O-S D-E P-R-U-E-B-A.A.A.A.A!!!

#include <iostream>
#include <cstdio>
#include <vector>
#include <queue>
#define forn(i, n) for(int i = 0; i < int(n); ++i)
using namespace std;

const int MAXN = 1024;

int N, M;
vector<int> G[MAXN];
vector<int> visitado;

int dfs(int v){
    visitado[v] = true;
    cout << "DFS - Estoy en " << v << endl;
    for(auto &w: G[v])
        if( not visitado[w])
             dfs(w);
}

void bfs(int v){
    queue<int> q;
    visitado[v] = true;
    q.push(v);
    while( not q.empty() ){
        v = q.front(); q.pop();
        cout << "BFS - Estoy en " << v << endl;
        for(auto &w: G[v])
            if( not visitado[w]){
                visitado[w] = true;
                q.push(w);
            }
    }
}

int main(){
    freopen("grafo.txt", "r", stdin);
    cin >> N >> M;
    forn(i, M){
        int u, v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }

    visitado.resize(N, false);
    dfs(0);

    visitado.clear();
    visitado.resize(N, false);
    bfs(0);

    return 0;
}
```

```cpp
#include <iostream>
#include <cstdio>
#include <vector>
#include <queue>
#include <algorithm>
#include <bitset>
#define forn(i, n) for(int i = 0; i < int(n); ++i)
using namespace std;

const int MAXN = 1024;


struct hedge{
    int v, d;
    bool operator<(const hedge &other) const {
        return d > other.d;
    }
};

int N, M;
vector<hedge> G[MAXN];
bitset<MAXN> done;
```

```cpp
vector<int> D, P;

void dijkstra(int r){
    priority_queue<hedge> q;
    D[r] = 0;
    q.push({r, 0});
    while( not q.empty() ){
        auto v = q.top().v; q.pop();
        if (done[v]) continue;
        done[v] = true;
        //cout << "La distancia de " << r << " a " << v.v << " es " <<
↪   v.d << endl;
        for(auto &w: G[v])
            if( D[w.v] == -1 or D[w.v] > D[v] + w.d) {
                D[w.v] = D[v] + w.d;
                P[w.v] = v;
                q.push({w.v, D[w.v]});
            }
    }
}

int main(){
    freopen("grafo.in", "r", stdin);

    cin >> N >> M;
    forn(i, M){
        int u, v, d;
        cin >> u >> v >> d;
        G[u].push_back({v, d});
        G[v].push_back({u, d}); // no dirigido
    }

    D.resize(N, -1);
    P.resize(N, -1);
    dijkstra(0);

    vector<int> camino;
    for(int v = 4; v != 0; v = P[v]) {
        camino.push_back(v);
    }
    reverse(camino.begin(), camino.end());
    cout << "Camino de 0 a 4:\n0";
    for(auto &v : camino)
        cout<< " -> " << v;

    return 0;
}
```

```cpp
#include <bits/stdc++.h>
using namespace std;
#define forr(i, a, b) for(int i = (a); i < (int) (b); i++)
#define forn(i, n) forr(i, 0, n)
#define dforr(i, a, b) for(int i = (int)(b-1); i >= (a); i--)
#define dforn(i, n) dforr(i, 0, n)
#define db(v) cerr << #v << " = " << v << endl
#define pb push_back
#define sz(x) ((int)x.size())
#define fst first
#define snd second
typedef long long ll;
typedef long double ld;
typedef pair<int, int> ii;
const int MAXN = 1050;

int n, m;
int B[MAXN][MAXN], C[MAXN][MAXN];
int LEFT[MAXN][MAXN], UP[MAXN][MAXN];

int mCuad(int i, int j){
    forn(k, MAXN){
        int ni = i + k, nj = j + k;
        if(ni >= n || nj >= m)return k;
        int reach = min(LEFT[ni][nj], UP[ni][nj]);
        if(reach < k)return k;
    }
    assert(false); // Si el algo llega acs porque codelgo como el tuje
}

int main(){
//        freopen("input.txt", "r", stdin);
//  ios :: sync_with_stdio(false); cin.tie(NULL);
```

```
    while(scanf("%d %d", &n, &m) >= 1){
        forn(i, n)forn(j, m)scanf("%d", &B[i][j]);
        forn(i, n)forn(j, m)scanf("%d", &LEFT[i][j]);
        forn(i, n)forn(j, m)scanf("%d", &UP[i][j]);

        forn(i, n)forn(j, m)
            C[i][j] = mCuad(i, j);
    }
        return 0;
}
```

```
#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast,unroll-loops")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define dbg(x) cerr << #x << " = " << x << endl;
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef long double ld;
typedef pair<int,int> ii;

vi KMPcompute (string s) { // saco los bordes, prefijos que son sufijos
    int N = (int)s.size();
    vi KMP(N,0);
    forsn(i,1,N) {
        int j = KMP[i-1];
        while (j and s[i] != s[j])
            j = KMP[j-1];
        if (s[i] == s[j]) j++;
        KMP[i] = j;
    }
    return KMP;
}

int main() {
    FAST_IO;

    string s; cin >> s;

    int ind = (int)s.size();
    vi show, border = KMPcompute(s);
    while (ind and border[ind-1]) {
        ind = border[ind-1];
        show.pb(ind);
    }

    dforn(i,show.size()) cout << show[i] << ' ';
}
```

```
// AC -> UVa 10405 (Online Judge)

#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef pair<int,int> ii;
```

```cpp
const int MAXN = 1005;

int dp[MAXN][MAXN];

int main() {
    FAST_IO;

    string a,b;
    while (getline(cin,a) and getline(cin,b)) {
        forn(i,a.size()+1) dp[i][0] = 0; // reset
        forn(i,b.size()+1) dp[0][i] = 0;
        forsn(i,1,a.size()+1) forsn(j,1,b.size()+1)
            dp[i][j] =
↪   max({dp[i-1][j],dp[i][j-1],(dp[i-1][j-1]+1)*(a[i-1] == b[j-1])});
        cout << dp[(int)a.size()][(int)b.size()] << '\n';
    }

    return 0;
}
```

```cpp
#include <bits/stdc++.h>
using namespace std;
#define sz(x) ((int)x.size())

const int MAXN = 1005;
vector<int> g1[MAXN], g2[MAXN], g3[MAXN];
queue<int> Q0, Q1, Q2, Q3;
void bfs(int st){

    while(sz(Q0) || sz(Q1) || sz(Q2) || sz(Q3)){
        while(sz(Q0)){
            int p = Q0.front(); Q0.pop();
            // do stuff...
            for(int q : g1[p]){ // Los vecinos a distancia 1.
                // blabla..
                Q1.push(q);
            }
            for(int q : g2[p]){ // Los vecinos a distancia 2.
                // blabla..
                Q2.push(q);
            }
            for(int q : g3[p]){ // Los vecinos a distancia 3.
                // blabla..
                Q3.push(q);
            }
        }
        Q0 = Q1; Q1 = Q2; Q2 = Q3; Q3 = {};
    }
}
int main(){
        return 0;
}
```

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

#define forn(i,n) for(int i = 0; i < n; i++)
#define forsn(i,s,n) for(int i = s; i < n; i++)
#define dforn(i,n) for(int i = n-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = n-1; i >= s; i--)

#define MAXN 1000

int valores[MAXN];

void merge (int l, int r, int m, int *arr) {
    int nl = m-l+1, nr = r-m;
    int larr[nl], rarr[nr];

    forn (i,nl) larr[i] = arr[l+i];
    forn (i,nr) rarr[i] = arr[m+i+1];

    int k = l, i = 0, j = 0;
    while (i < nl && j < nr)
        if (larr[i] <= rarr[j]) arr[k++] = larr[i++];
        else arr[k++] = rarr[j++];

    while (i < nl) arr[k++] = larr[i++];
```

```
    while (j < nr) arr[k++] = rarr[j++];
}

void divide (int l, int r, int *arr) {
    if (l >= r) return;

    int mid = l + (r-l)/2;
    divide(l,mid,arr);
    divide(mid+1,r,arr);

    merge(l,r,mid,arr);
}

int main() {
    int N; scanf("%d",&N);
    forn (i,N) scanf("%d",&valores[i]);

    divide(0,N-1,valores);

    forn (i,N) printf("%d ",valores[i]);

    return 0;
}

// AC -> Aizu CGL 4B

#include <bits/stdc++.h>

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef pair<int,int> ii;

const double EPS = 1e-6; // margin of error

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

    double norm2() {return (*this) * (*this);} // modulo of vector,
↪   squared
    double norm() {return sqrt(norm2());}
    pt operator* (const double &t) const {return pt(x*t,y*t);}
    pt operator- (const pt &o) const {return pt(x-o.x, y-o.y);}
    double operator% (const pt &o) const {return x*o.y - y*o.x;} //
↪   cross product
    double operator* (const pt &o) const {return x*o.x + y*o.y;} // dot
↪   product
    bool operator< (const pt &o) const { // chull sort cmp
        return (x < o.x || (x == o.x && y < o.y));
    }
    bool left (const pt &a, const pt &b) {return (b-a)%(*this-a) >
↪   EPS;}
};

vector<pt> chull (vector<pt> &p) {
    vector<pt> r;
    sort(all(p));
    forn (i,p.size()) { // lower hull
        while (r.size() >= 2 and r.back().left(r[r.size()-2],p[i]))
↪   r.pop_back();
        r.pb(p[i]);
    }
    r.pop_back(); int k = r.size();
    dforn (i,p.size()) { // upper hull
        while (r.size() >= k+2 and r.back().left(r[r.size()-2],p[i]))
↪   r.pop_back();
        r.pb(p[i]);
    }
    r.pop_back();
    return r;
```

```
}

struct poly {
    int n; vector<pt> p;
    poly(){}
    poly(vector<pt> _p) {p=_p; n=p.size();}

    double callipers() { // returns square of max dist
        double r = 0;
        int j = (n >= 2); // doesn't exist if there's no such pair of
↪   points
        forn (i,j) {
            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;
    }
};

vector<pt> p;

int main() {
    FAST_IO;

    int n; cin >> n;

    forn (i,n) {
        double x,y; cin >> x >> y;
        p.pb({x,y});
    }

    cout << setprecision(10) << sqrt(poly(chull(p)).callipers());

    return 0;
}

/// ESCRIBen vez de tanto dar vueltas
/// si te parece que no va PROBALO PRIMERO!
```

```
/// CODEA LO BSICO PRIMERO!
/// HACE C-A-S-O-S D-E P-R-U-E-B-A.A.A.A.A!!!

#include <bits/stdc++.h>
using namespace std;
#define forr(i, a, b) for(int i = (a); i < (int) (b); i++)
#define forn(i, n) forr(i, 0, n)
#define dforr(i, a, b) for(int i = (int)(b-1); i >= (a); i--)
#define dforn(i, n) dforr(i, 0, n)
#define db(v) cerr << #v << " = " << v << endl
#define pb push_back
#define sz(x) ((int)x.size())
#define fst first
#define snd second
typedef long long ll;
typedef long double ld;
typedef pair<int, int> ii;
const int MAXN = -1;

ld fun(ld t){ return sqrtl((t-4)*(t-4));}

// Todas estas son para hallar el mmo
// Si querel mmo cambil sentido del operador en el if()

// Quite unpopular, pero similar a BS()
ll TS(ll l, ll r){
    while(r - l > 2){
        ll tl = (2*l + r)/3, tr = (l + 2*r)/3;
        if(fun(tl) < fun(tr))r = tr;
        else l = tl;
    }
    while(fun(l+1) < fun(l))l++; // Prueba a lo sumo 3 veces
    return l;
}

// Igual al anterior pero en flotantes funciona mucho mejor
// Por lo tanto en este caso es popular
ld TSD(ld l, ld r){
    int cnt = 50; // Ajustable si estjugado con el tiempo
    while(cnt--){
        ld tl = (2*l + r)/3, tr = (l + 2*r)/3;
```

```
        if(fun(tl) < fun(tr))r = tr;
        else l = tl;
    }
    return l;
}

// Para enteros es mucho mpopular usar una BS() de la siguiente forma
ll BS(ll l, ll r){
    while(r - l > 1){ // Recordrocurar que la respuesta sea
↪   estrictamente mayor a l y menor a r;
        ll m = (l + r)/2;
        if(fun(m) <= fun(m+1))r = m;
        else l = m;
    }
    return r;
}


int n;

int main(){
//        freopen("input.txt", "r", stdin);
//  ios :: sync_with_stdio(false); cin.tie(NULL);
//    while(scanf("%d", &n) >= 1){
    printf("Con la primer ternary: %lld\n", TS(0, 10));
    printf("Con la segunda ternary: %.20Lf\n", TSD(0.0, 10.0));
    printf("Con la binary: %lld\n", BS(0, 10));
//    }
        return 0;
}

#include <iostream>
#include <string>
#include <map>
using namespace std;

struct trie{
    map<char, trie> edges;
    int count = 0;
    void insert(const string &s, int i=0){
        if( s.length() == i ) count++;
        else edges[s[i]].insert(s, i+1);
    }
    void print(const string &&s=""){
        if( count > 0 ) cout << s << endl;
        for(auto &e: edges)
            e.second.print(s+e.first);
    }
};

int main(){

    trie t;

    t.insert("hola");
    t.insert("horno");
    t.insert("holas");

    t.print();

    return 0;
}

// AC -> CSES - Giant Pizza

#include <bits/stdc++.h>

#pragma GCC optimize("Ofast")
#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
```

```cpp
typedef long long ll;
typedef pair<int,int> ii;

const int MAXN = 2e5+5;
const int INF = 1e9+5;

void fastscan (int &x) {
    int c; x = 0;
    c=getchar_unlocked();
    for(; c>='0' && c <='9'; c=getchar_unlocked())
        x = 10*x + c-'0';
}

vi G[MAXN];
int stTime[MAXN], col[MAXN], posTopSort[MAXN];
bitset<MAXN> matched, rta;
int prox_libre = 1, actTime = 0, pos = 0;

vi pila;
int tarjan (int st) { // SCC con Tarjan
    stTime[st] = actTime++;
    pila.pb(st);

    int mini = actTime-1;
    for (auto &i : G[st]) {
        if (stTime[i] == -1) mini = min(mini,tarjan(i));
        if (!matched[i]) mini = min(mini,stTime[i]);
    }

    if (mini >= stTime[st])
        while (not pila.empty()) {
            int e = pila.back(); pila.pop_back();
            col[e] = prox_libre; matched[e] = true;

            // si estn en la misma componente los opuestos chau, as que
            // no me importa que meta esto de la pila y lo cuente al
↪   revs
            // y directamente me guardo posiciones hipotticas as puedo
↪   comparar
            // de a pares
            posTopSort[e] = pos++;
            if (e == st) {prox_libre++; break;}
        }

    return mini;
}

int main() {
    memset(stTime,-1,sizeof(stTime)); // reset, si no tengo tiempo
↪   inicio

    int n,m; fastscan(n); fastscan(m);
    forn(i,n) { // duplicar es +, de lo contrario -
        char s1,s2; int x1,x2;
        s1=getchar_unlocked(); getchar_unlocked();
        fastscan(x1);
        s2=getchar_unlocked(); getchar_unlocked();
        fastscan(x2);
        x1--; x2--;

        x1 *= 2; x2 *= 2; // duplico as tengo mis opuestos, adyacentes
↪   en el array
        // +1 si opuesto es positivo, osea si es negativo. Y x2 +1 si
↪   es positivo
        G[x1+(s1 == '-')].pb(x2+(s2 == '+'));
        // +1 x2 si opuesto es positivo, osea si es negativo. Y x1 +1
↪   si es positivo
        G[x2+(s2 == '-')].pb(x1+(s1 == '+'));
    }

    forn(i,2*m) if (stTime[i] == -1) tarjan(i);

    bool posib = true;
    for (int i = 0; i < 2*m; i += 2) { // comparo adyacentes, salto de
↪   a 2
        if (col[i] == col[i+1]) {posib = false; break;} // y si estn en
↪   la misma SCC, chau
        // si lo puse primero, est debajo en el topSort, y a es
↪   verdadero si est despus de a
        // es decir, al hacer el reverse con Tarjan, la posA < posA.
        rta[(i+1)/2] = (posTopSort[i+1] < posTopSort[i]);
```

```
    }

    if (!posib) printf("IMPOSSIBLE");
    else forn(i,m) {putchar_unlocked(rta[i] ? '+' : '-');
↪   putchar_unlocked(' ');}

    return 0;
}

#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast,unroll-loops")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define dbg(x) cerr << #x << " = " << x << endl;
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
typedef long double ld;
typedef pair<int,int> ii;

const int MAXN = 1e5+5;
const int MAXM = 2*MAXN;

struct node {
    int v,id;
};

void fs (int &x) {
    int c; x = 0;
    c = getchar_unlocked();
    if (c < '0' || c > '9') c = getchar_unlocked();
    for (; c >= '0' && c <= '9'; c = getchar_unlocked())
        x = 10*x + c - '0';
}

vector<node> G[MAXN];
vector<ii> rta;
int tin[MAXN], actT = 1;
bitset<MAXM> matched;
bool posib = true;

vector<pair<ii,int>> pila; // (a -> b), id
int tarjan(int st) {
    tin[st] = actT++;

    int mini = tin[st];
    for (auto &i : G[st]) {
        if (!matched[i.id]) {
            int aux = tin[st];
            matched[i.id] = true;
            pila.pb({{st,i.v},i.id});

            if (!tin[i.v]) aux = min(aux,tarjan(i.v));
            if (tin[i.v] > 0) aux = min(aux,tin[i.v]);

            if (aux >= tin[st]) {
                int cnt = 0;
                while (!pila.empty()) {
                    auto e = pila.back(); pila.pop_back();
                    cnt++;
                    rta.pb(e.fst);
                    if (e.snd == i.id) break; // hasta el mío incluído
                }

                if (cnt == 1) posib = false; // no se puede si hay
↪   puentes
            }

            mini = min(aux,mini);
        }
    }
```

```
    return mini;
}

int main() {
    int n,m; fs(n), fs(m);

    forn(i,m) {
        int u,v; fs(u), fs(v); u--, v--;
        G[u].pb({v,i}), G[v].pb({u,i});
    }

    tarjan(0);
    if (posib && (int)rta.size() < m) posib = false; // nunca me dice
↪   que sea un grafo conexo

    if (posib)
        for (auto &i : rta) printf("%d %d\n",i.fst+1,i.snd+1);
    else printf("IMPOSSIBLE");

    return 0;
}
```

```
#include <bits/stdc++.h>

//#pragma GCC optimize("Ofast")
//#pragma GCC target("avx,avx2,fma")

#define forn(i,n) for(int i = 0; i < int(n); i++)
#define forsn(i,s,n) for(int i = int(s); i < int(n); i++)
#define dforn(i,n) for (int i = int(n)-1; i >= 0; i--)
#define dforsn(i,s,n) for(int i = int(n)-1; i >= int(s); i--)
#define all(c) (c).begin(),(c).end()
#define pb push_back
#define fst first
#define snd second
#define FAST_IO ios::sync_with_stdio(false);cin.tie(nullptr);

using namespace std;
typedef vector<int> vi;
typedef long long ll;
```

```
typedef pair<int,int> ii;

const int MAXN = 2e5+2;

void fs (int &x) {
    int c; x = 0;
    c = getchar_unlocked();
    if (c < '0' || c > '9') c = getchar_unlocked();
    for (; c>='0' && c<='9'; c = getchar_unlocked())
        x = 10*x + c-'0';
}

void fp (int x) {
    int i = 9;
    char buf[10];
    while (x) buf[i--] = (x%10)+'0', x /= 10;
    while ((++i) < 10) putchar_unlocked(buf[i]);
}

int FT[MAXN];

void setFT (int p, int v) {
    for (int i = p; i < MAXN; i += i & -i)
        FT[i] += v;
}

int getFT (int p) {
    int r = 0;
    for (int i = p; i; i -= i & -i)
        r += FT[i];
    return r;
}

int invertFT (int v) {
    int x = 0;
    for (int d = (1<<(31-__builtin_clz(MAXN))); d; d >>= 1)
        if ((x|d) < MAXN && FT[x|d] < v) x |= d, v -= FT[x];
    return x+1;
}

int main() {
```

```
    int n,k; fs(n), fs(k);
    if (n == 1) return printf("1"), 0;

    forn(j,n) setFT(j+1,1);

    int ind = 1, myN = n+1;
    forn(i,n) {
        ind = (ind+k-1)%(--myN)+1;
        int v = invertFT(ind);
        fp(v); putchar_unlocked(' ');
        setFT(v,-1);
    }

    return 0;
}
```