Team notebook

November 7, 2024

Contents

1 Geometry

1.1 GRAHAM

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

using namespace std;
#define pb push_back
#define f first
#define s second
#define int int64_t
#define pi pair<int,int>


pi operator-(const pi &l, const pi &r) { return {l.f -
    r.f, l.s - r.s}; }
int norm(const pi &p) { return (p.f*p.f) + (p.s*p.s); } //
    x^2 + y^2
int cross(const pi &a, const pi &b) { return a.f * b.s -
    a.s * b.f; } // cross product
int cross(const pi &p, const pi &a, const pi &b) {
                    // cross product
        return cross(a - p, b - p);
}

vector<int> hullInd(const vector<pi> &v) {
    if(v.size()==0)return {};
        int ind = int(min_element(v.begin(),v.end()) -
            v.begin());
        vector<int> cand, hull{ind};
        for(int i=0;i<v.size();i++) if (v[i] != v[ind])
            cand.pb(i);

        sort(cand.begin(),cand.end(), [&](int a, int b) {
            // sort by angle, tiebreak by distance
            pi x = v[a] - v[ind], y = v[b] - v[ind];
            int t = cross(x, y);
            return t != 0 ? t > 0 : norm(x) < norm(y);
        });

        for(int c : cand) { // for every point
            while (hull.size() > 1 &&
                    cross(v[end(hull)[-2]],
                    v[hull.back()], v[c]) <= 0) {
                    hull.pop_back(); // pop until
                        counterclockwise and size > 1
            }
            hull.pb(c);
        }

    return hull;
}

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

    #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif

    int t; cin >> t;
    while(t--){
        vector<pi> v;
        int n; cin >> n;
        for(int i=0;i<n;i++){
            int a; int b; cin >> a >> b;
            v.pb({a,b});
        }

        vector<int> ans = hullInd(v);
        cout << ans.size() << "\n";
        for(int i : ans){
            cout << v[i].f << " " << v[i].s << "\n";
        }
    }


}
```

1.2 HullDiameter

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

using namespace std;
#define pb push_back
#define f first
#define s second
#define int int64_t
#define pi pair<int,int>
```

```cpp
pi operator-(const pi &l, const pi &r) { return {l.f -
    r.f, l.s - r.s}; }
int norm(const pi &p) { return (p.f*p.f) + (p.s*p.s); } //
    x^2 + y^2
int cross(const pi &a, const pi &b) { return a.f * b.s -
    a.s * b.f; } // cross product
int cross(const pi &p, const pi &a, const pi &b) {
                // cross product
    return cross(a - p, b - p);
}
double dist(pi &a, pi &b){
    pi d = a-b; return sqrt(norm(d));
}
// db diameter2(vP P) {
//     P = hull(P);
//     int n = sz(P), ind = 1; T ans = 0;
//     if (n > 1) FOR(i,n) for (int j = (i+1)%n;;ind =
    (ind+1)%n) {
//             ckmax(ans, abs2(P[i]-P[ind]));
//             if (cross(P[j]-P[i],P[(ind+1)%n]-P[ind]) <=
    0) break;
//     }
//     return ans;
// }
// unchecked
double diameter2(vector<pi> v) {
        int n = v.size(), ind = 1; double ans = 0;
        if (n > 1) for(int i=0;i<n;i++) for (int j =
            (i+1)%n;;ind = (ind+1)%n) {
        if(dist(v[i],v[ind]) > ans){
            ans = dist(v[i],v[ind]);
        }
                if (cross(v[j]-v[i],v[(ind+1)%n]-v[ind]) <=
                    0) break;
        }
        return ans;
}
signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif


}
```

1.3   LineContainer

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int int64_t
struct Line {
        mutable int k, m, p;
        bool operator<(const Line& o) const { return k <
            o.k; }
        bool operator<(int x) const { return p < x; }
```

```cpp
};
struct LineContainer : multiset<Line, less<>> {
        // (for doubles, use inf = 1/.0, div(a,b) = a/b)
        static const int inf = LLONG_MAX;
        int div(int a, int b) { // floored division
                return a / b - ((a ^ b) < 0 && a % b); }
        bool isect(iterator x, iterator y) {
                if (y == end()) return x->p = inf, 0;
                if (x->k == y->k) x->p = x->m > y->m ? inf :
                    -inf;
                else x->p = div(y->m - x->m, x->k - y->k);
                return x->p >= y->p;
        }
        void add(int k, int m) {
                auto z = insert({k, m, 0}), y = z++, x = y;
                while (isect(y, z)) z = erase(z);
                if (x != begin() && isect(--x, y)) isect(x,
                    y = erase(y));
                while ((y = x) != begin() && (--x)->p >=
                    y->p)
                        isect(x, erase(y));
        }
        int query(int x) {
                assert(!empty());
                auto l = *lower_bound(x);
                return l.k * x + l.m;
        }
};
```

1.4   MONOTONEHULL

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

using namespace std;
#define pb push_back
#define f first
#define s second
#define int int64_t
#define pi pair<int,int>


pi operator-(const pi &l, const pi &r) { return {l.f -
    r.f, l.s - r.s}; }
int norm(const pi &p) { return (p.f*p.f) + (p.s*p.s); } //
    x^2 + y^2

int cross(const pi &a, const pi &b) { return a.f * b.s -
    a.s * b.f; } // cross product
int cross(const pi &p, const pi &a, const pi &b) {
                        // cross product
        return cross(a - p, b - p);
}

vector<pi> hull;
vector<pi> points;
void monotone_chain() {
        // sort with respect to the x and y coordinates
```

```cpp
        sort(points.begin(), points.end());
        // distinct the points
        points.erase(unique(points.begin(), points.end()),
            points.end());
        int n = points.size();

        // 1 or 2 points are always in the convex hull
        if (n < 3) {
                hull = points;
                return;
        }

        // lower hull
        for (int i = 0; i < n; i++) {
                // if with the new point points[i], a right
                    turn will be formed,
                // then we remove the last point in the hull
                    and test further
                while (hull.size() > 1 &&
                    cross(hull[hull.size() - 2],
                        hull.back(), points[i]) <= 0)

                        hull.pop_back();
                // otherwise, add the point to the hull
                hull.push_back(points[i]);
        }

        // upper hull, following the same logic as the
            lower hull
        auto lower_hull_length = hull.size();
        for (int i = n - 2; i >= 0; i--) {
                // we can only remove a point if there are
                    still points left in the
                // upper hull
                while (hull.size() > lower_hull_length &&
                    cross(hull[hull.size() - 2],
                        hull.back(), points[i]) <= 0)
                        hull.pop_back();
                hull.push_back(points[i]);
        }
        // delete point[0] that has been added twice
        hull.pop_back();
}

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

    // #ifndef ONLINE_JUDGE
    // freopen("file.txt", "r", stdin);
    // #endif

    int t; cin >> t;
    while(t--){
        int n; cin >> n;
        points.clear();
        hull.clear();
        for(int i=0;i<n;i++){
            int a; int b; cin >> a >> b;
            points.pb({a,b});
        }
```

```cpp
        monotone_chain();
        cout << hull.size() << "\n";
        for(pi i : hull){
            cout << i.f << " " << i.s << "\n";
        }
        //cout << "\n---------\n";
    }


}
```

1.5 PointToStandardForm

```cpp
#include <bits/stdc++.h>
using namespace std;
#define f first
#define s second
#define pi pair<int,int>
//tested very little
pair<pi, int> get_line(pi a, pi b) {
        pi z = {b.f - a.f, b.s - a.s};
        swap(z.f, z.s);

        z.f *= -1;
        int g = __gcd(z.f, z.s);
        z.f /= g;
        z.s /= g;

        z = max(z, {-z.f, -z.s});
        return {z, z.f * a.f + z.s * a.s};
}
```

1.6 SegmentIntersection

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

using namespace std;
#define pb push_back
#define f first
#define s second
#define int int64_t
#define pi pair<int,int>


pi operator-(const pi &l, const pi &r) { return {l.f -
    r.f, l.s - r.s}; }
int norm(const pi &p) { return (p.f*p.f) + (p.s*p.s); } //
    x^2 + y^2
int cross(const pi &a, const pi &b) { return a.f * b.s -
    a.s * b.f; } // cross product
int cross(const pi &p, const pi &a, const pi &b) { //
    cross product
    return cross(a - p, b - p);
}
```

```cpp
int sn(int x){
    if (x == 0) return 0;
    return x/abs(x);
}
bool rect_int(pi p1, pi p2, pi p3, pi p4) {
        int x1, x2, x3, x4, y1, y2, y3, y4;
        x1 = min(p1.f, p2.f), x2 = max(p1.f, p2.f);
        y1 = min(p1.s, p2.s), y2 = max(p1.s, p2.s);
        x3 = min(p3.f, p4.f), x4 = max(p3.f, p4.f);
        y3 = min(p3.s, p4.s), y4 = max(p3.s, p4.s);
        return !(x2 < x3 || x4 < x1 || y2 < y3 || y4 < y1);
}

bool segmentIntersect(pi p1, pi p2, pi p3, pi p4){
    return (rect_int(p1,p2,p3,p4) && sn(cross(p1,p2,p4)) *
        sn(cross(p1,p2,p3)) <= 0 && sn(cross(p3,p4,p1)) *
        sn(cross(p3,p4,p2)) <= 0);
}

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

    #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif
    int t; cin >> t;
    while(t--){
        int x1; int y1; int x2; int y2; int x3; int y3; int
            x4; int y4;
        cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> x4 >> y4;
        if(segmentIntersect({x1,y1},{x2,y2},{x3,y3},{x4,y4})){
            cout << "YES";
        } else {
            cout << "NO";
        }
        cout <<"\n";


    }


}
```

1.7 Shoelace

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

using namespace std;
#define pb push_back
#define f first
#define s second
#define int int64_t
#define pi pair<int,int>

//area is signed
int twiceArea(vector<pi>pts){
    int ans = 0;
```

```cpp
    for(int i=0;i<pts.size()-1;i++){
        ans += pts[i].f * pts[i+1].s - pts[i].s *
            pts[i+1].f;
    }
    ans += pts.back().f *pts[0].s - pts.back().s * pts[0].f;
    return ans;
}
signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif
    int n; cin >> n;
    vector<pi> pts;
    for(int i=0;i<n;i++){
        int a; int b; cin >> a >> b; pts.pb({a,b});
    }
    cout << abs(twiceArea(pts)); // maybe use labs here
}
```

2 Graphs

2.1 DINIC

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


typedef long long ll;

using namespace std;
//using namespace __gnu_pbds;
#define ordered_set tree<int, null_type,less<int>,
    rb_tree_tag,tree_order_statistics_node_update>
#define pb push_back
#define f first
//#define s second
//#define int ll
#define pi pair<int,int>
#define pf pair<float,float>

struct Dinic {   // flow template
    using F = ll; // flow type
    struct Edge {
        int to;
        F flo, cap;
    };
    int N;
    vector<Edge> eds;
    vector<vector<int>> adj;
    void init(int _N) {
        N = _N;
        adj.resize(N), cur.resize(N);
    }
    /// void reset() { trav(e,eds) e.flo = 0; }
    void ae(int u, int v, F cap, F rcap = 0) {
        assert(min(cap, rcap) >= 0);
```

```cpp
                adj[u].pb((eds).size());
                eds.pb({v, 0, cap});
                adj[v].pb(eds.size());
                eds.pb({u, 0, rcap});
        }
        vector<int> lev;
        vector<vector<int>::iterator> cur;
        bool bfs(int s, int t) { // level = shortest
            distance from source
                lev = vector<int>(N, -1);
                for(int i=0;i<N;i++) cur[i] = begin(adj[i]);
                queue<int> q({s});
                lev[s] = 0;
                while (q.size()) {
                        int u = q.front();
                        q.pop();
                        for (auto e : adj[u]) {
                                const Edge &E = eds[e];
                                int v = E.to;
                                if (lev[v] < 0 && E.flo <
                                    E.cap) q.push(v), lev[v]
                                    = lev[u] + 1;
                        }
                }
                return lev[t] >= 0;
        }
        F dfs(int v, int t, F flo) {
                if (v == t) return flo;
                for (; cur[v] != end(adj[v]); cur[v]++) {
                        Edge &E = eds[*cur[v]];
                        if (lev[E.to] != lev[v] + 1 || E.flo
                            == E.cap) continue;
                        F df = dfs(E.to, t, min(flo, E.cap -
                            E.flo));
                        if (df) {
                                E.flo += df;
                                eds[*cur[v] ^ 1].flo -= df;
                                return df;
                        } // saturated >=1 one edge
                }
                return 0;
        }
        F maxFlow(int s, int t) {
                F tot = 0;
                while (bfs(s, t))
                        while (F df = dfs(s, t,
                            numeric_limits<F>::max())) tot
                            += df;
                return tot;
        }
};

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

    // #ifndef ONLINE_JUDGE
    // freopen("file.txt", "r", stdin);
    // #endif
    int l; int r; int n; cin>>l>>r>>n;
    Dinic d;
    d.init(l+r+2);
```

```cpp
    for(int i=0;i<n;i++){
        int a; int b; cin>>a>>b;
        d.ae(a+1,l+b+1,1);
    }
    for(int i=0;i<l;i++){
        d.ae(0,i+1, 1);
    }
    for(int i=0;i<r;i++){
        d.ae(i+1+l,l+r+1, 1);
    }

    cout<< d.maxFlow(0, l+r+1)<< "\n";
    d.bfs(0,l+r+1);

    for(int i=1;i<=l;i++){
        for(int v:d.adj[i]){
            if(d.eds[v].cap==0) continue;
            if(d.eds[v].cap==d.eds[v].flo) cout<< i-1 <<"
                "<< d.eds[v].to-l-1<<"\n";
        }
    }
    return 0;


}
```

2.2 DSU by size

```cpp
#include<bits/stdc++.h>
using namespace std;
int parent[1];//fill
int sz[1]; //fill
void make_set(int v) {
    parent[v] = v;
    sz[v] = 1;
}
int find_set(int v) {
    if (v == parent[v])
        return v;
    return parent[v] = find_set(parent[v]);
}
void union_sets(int a, int b) {
    a = find_set(a);
    b = find_set(b);
    if (a != b) {
        if (sz[a] < sz[b])
            swap(a, b);
        parent[b] = a;
        sz[a] += sz[b];
    }
}
```

2.3 HUNGRY

```cpp
/*
```

```cpp
ID: sahajrastogi
LANG: C++11
*/

#include <iostream>
#include <bits/stdc++.h>
#include <unordered_set>
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/tree_policy.hpp>

typedef long long ll;

using namespace std;
//using namespace __gnu_pbds;
#define ordered_set tree<int, null_type,less<int>,
    rb_tree_tag,tree_order_statistics_node_update>
#define pb push_back
#define pi pair<int,int>
#define f first
#define s second
#define int int64_t


int ckmin(int &a, int b) { return a > b ? ((a = b), true)
    : false; }

/**
 * @return the jobs of each worker in the optimal
     assignment,
 * or -1 if the worker is not assigned
 */
template <class T> vector<int> hungarian(const
    vector<vector<T>> &C) {
        int J = C.size();
        int W = C[0].size();
        assert(J <= W);

        // job[w] = job assigned to w-th worker, or -1 if
            no job assigned
        // note: a W-th worker was added for convenience
        vector<int> job(W + 1, -1);
        vector<T> h(W); // Johnson potentials

        const T inf = numeric_limits<T>::max();
        // assign j_cur-th job using Dijkstra with
            potentials
        for (int j_cur = 0; j_cur < J; j_cur++) {
                int w_cur = W; // unvisited worker with
                    minimum distance
                job[w_cur] = j_cur;

                vector<T> dist(W + 1, inf); //
                    Johnson-reduced distances
                dist[W] = 0;
                vector<bool> vis(W + 1);  // whether visited
                    yet
                vector<int> prv(W + 1, -1); // previous
                    worker on shortest path
                while (job[w_cur] != -1) { // Dijkstra step:
                    pop min worker from heap
                    T min_dist = inf;
                    vis[w_cur] = true;
```

```cpp
            int w_next = -1; // next unvisited
                    worker with minimum distance


            // consider extending shortest path
                    by w_cur -> job[w_cur] -> w
            for (int w = 0; w < W; w++) {
                    if (!vis[w]) {
                            // sum of reduced edge
                                    weights w_cur ->
                                    job[w_cur] -> w
                        T edge =
                                C[job[w_cur]][w]
                                - h[w];
                        if (w_cur != W) {
                                edge -=
                                        C[job[w_cur]][w_cur]
                                        - h[w_cur];
                                assert(edge >=
                                        0);
                        }
                        if (ckmin(dist[w],
                                dist[w_cur] +
                                edge)) { prv[w] =
                                w_cur; }
                        if (ckmin(min_dist,
                                dist[w])) {
                                w_next = w; }
                    }
            }
            w_cur = w_next;
        }

        for (int w = 0; w < W; w++) { // update
            potentials
                ckmin(dist[w], dist[w_cur]);
                h[w] += dist[w];
        }

        while (w_cur != W) { // update job assignment
                job[w_cur] = job[prv[w_cur]];
                w_cur = prv[w_cur];
        }
    }

    return job;
}

signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
        #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif
    int n;
    cin>> n;
    vector<vector<int>> table(n, vector<int>(n));
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            cin>>table[j][i];
        }
    }
    vector<int> sol = hungarian(table);
```

```cpp
        int cost=0;
        for(int i=0;i<n;i++) cost+=table[sol[i]][i];
        cout<< cost<<"\n";
        for(int i=0;i<n;i++){
            cout << sol[i]+1<<" "<< i+1;
            cout<< "\n";
        }
}
```

2.4 KOSARAJU

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


typedef long long ll;

using namespace std;
//using namespace __gnu_pbds;
#define ordered_set tree<int, null_type,less<int>,
        rb_tree_tag,tree_order_statistics_node_update>
#define pb push_back
#define f first
//#define s second
//#define int ll
#define pi pair<int,int>
#define pf pair<float,float>

vector<int> adj[500005];
vector<int> adjr[500005];
int visited[500005]={0};
vector<int> order;
vector<int> scc[500005];
int k = 0;

void dfs(int x){
    visited[x] = 1;
    for(auto nex : adj[x]){
        if(!visited[nex])dfs(nex);
    }
    order.push_back(x);
}

void dfsr(int x){
    visited[x] = k;
    scc[k].pb(x);
    for(auto nex : adjr[x]){
        if(!visited[nex]) dfsr(nex);
    }
}
signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    // #ifndef ONLINE_JUDGE
    // freopen("file.txt", "r", stdin);
    // #endif

    int n; int m; cin>> n >> m;
    for(int i=0;i<m;i++){
```

```cpp
        int a; int b; cin >> a >> b;
        adj[a].pb(b);
        adjr[b].pb(a);
    }

    k=0;

    for(int i=0;i<n;i++){
        if(!visited[i]) dfs(i);
    }

    reverse(order.begin(),order.end());
    for(int i=0;i<500003;i++) visited[i]=0;

    for(int x : order){
        if(!visited[x]){
            k++;
            dfsr(x);
        }
    }

    cout << k << "\n";
    for(int i=1;i<=k;i++){
        cout << scc[i].size();
        for(auto x : scc[i]){
            cout << " "<<x;
        }
        if(i!=k) cout << "\n";
    }

}
```

3 Matrix

3.1 MATRIX

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int int64_t
#define f first
#define s second
const int MN = 505;
const int mod = 998244353;


int power(int b, int e, int m){
    if(e >= 1){
        int p = power(b, e / 2, m) % m;
        if(e%2==0){
            return (p*p)%m;
        } else {
            return (b*((p*p)%m)%m);
        }
    } else if(e == 1) {
        return (b%m);
    } else {
        return 1;
```

```cpp
        }
    }
    int inv(int b, int m){
        return power(b,m-2,m);
    }
    struct matrix {
        int r, c;
        double m[MN][MN];
        matrix (int _r, int _c) : r (_r), c (_c) {
            memset(m, 0, sizeof m);
        }
        void print() {
            for (int i = 0; i < r; ++i) {
                for (int j = 0; j < c; ++j)
                    cout << m[i][j] << " ";
                cout << endl;
            }
        }

        matrix operator *(const matrix &b){
            matrix res(r, b.c);
            if(c!=b.r) cout<< "bad matrix multiplication";
            for(int i=0;i<r;i++){
                for(int j=0;j<b.c;j++){
                    for(int k=0;k<c;k++){
                        res.m[i][j]+=m[i][k]*b.m[k][j];
                        //res.m[i][j]%=mod;
                    }
                }
            }
            return res;
        }

        void operator *=(const matrix &b){
            *this = *this * b;
            //return *this;
        }

        matrix operator ^(int e){
            matrix res(r,r);
            //matrix id(r,r);
            matrix b = *this;
            for (int i = 0; i < r; ++i)
                res.m[i][i] = 1;
            if (e == 0) return res;
            while (true) {
                if (e & 1) res *= b;
                if ((e >>= 1) == 0) break;
                b *= b;
            }
            return res;
        }

        void operator ^=(int e){
            *this = *this ^ e;
            //return *this;
        }
    };
    int getRow(vector<vector<int>>& m, int R, int i, int nex) {
        for(int j =nex; j<R;j++) if (m[j][i] != 0) return j;
        return -1; }
```

```cpp
    int getRow(vector<vector<double>>& m, int R, int i, int
        nex) {
        pair<double,int> bes{0,-1}; // find row with max
            abs value
        for(int j = nex; j< R;j++) bes =
            max(bes,{abs(m[j][i]),j});
        return bes.f < 1e-9 ? -1 : bes.s; }

    //for determinant and rank
    pair<int,int> gauss(vector<vector<int>> &m) { // convert
        to reduced row echelon form
        if (!m.size()) return {1,0};
        int R = m.size(), C = m[0].size(), rank = 0, nex =
            0;
        int det = 1; // determinant
        for(int i=0;i<C;i++) {
            int row = getRow(m,R,i,nex);
            if (row == -1) { det = 0; continue; }
            if (row != nex) det *= -1,
                swap(m[row],m[nex]);
            det *= m[nex][i]; rank++;
        det %= mod;
        //det = fmod(det,mod);
        //while(det < 0) det+= mod;
            int x = inv(m[nex][i],mod); for(int k = i;k
                < C;k++){
            m[nex][k] *= x;
            m[nex][k] %= mod;
        }
                for(int j=0;j < R;j++) if (j != nex) {
                    int v = m[j][i]; if (v == 0) continue;
                    for(int k=i;k<C;k++){
                    m[j][k] -= v*m[nex][k];
                    m[j][k] %= mod;
                    //m[j][k] = fmod(m[j][k],mod);
                }
            }
            nex++;
        }
        // for(int i = 0; i<R;i++){
        //     for(int j=0;j<C;j++){
        //         if(m[i][j] != 0){
        //             //rank++;
        //             break;
        //         }
        //     }
        // }
        return {det,rank};
    }

    //for system of linear equations with in double form
    void slae(vector<vector<double>> &m) { // convert to
        reduced row echelon form
        if (!m.size()) return;
        int R = m.size(), C = m[0].size(), nex = 0;
        for(int i=0;i<C;i++) {
            int row = getRow(m,R,i,nex);
            if (row == -1) { continue; }
            if (row != nex) swap(m[row],m[nex]);
            double x = 1/m[nex][i];
        for(int k = i;k < C;k++){
            m[nex][k] *= x;
```

```cpp
        }
            for(int j=0;j < R;j++) if (j != nex) {
                double v = m[j][i]; if (v == 0)
                    continue;
                for(int k=i;k<C;k++){
                m[j][k] -= v*m[nex][k];
            }
            }
            nex++;
        }
    }

    //returns -1 for no soln, 0 for 1 soln, and 1 for infinite
    int checkSoln(vector<vector<double>> &m){
        int r = m.size(); int c= m[0].size();
        int cnt = 0; bool imp = false; bool broke = false;
        for(int i = 0;i<r;i++){
            broke = false;
            for(int j = 0;j<c-1;j++){
                if(abs(m[i][j]) > 1e-9) {
                    cnt++; broke = true; break;
                }
            }
            if(!broke && abs(m[i][c-1]) > 1e-9) imp = true;
        }
        //cout << m[r-1][c-1] << "! ";
        //cout << cnt << "? ";

        if(imp) return -1;
        if(cnt < c-1) return 1;
        return 0;
    }
    signed main(){
        ios_base::sync_with_stdio(false);
        cin.tie(NULL);

        #ifndef ONLINE_JUDGE
        freopen("file.txt", "r", stdin);
        #endif
        while(true){
            int n; cin >> n; if(n==0)break;

            vector<vector<double>>mat(n,vector<double>(n+1,0));
            for(int i=0;i<n;i++){
                for(int j=0;j<n;j++){
                    cin >> mat[i][j];
                }
            }
            for(int i=0;i<n;i++){
                cin >> mat[i][n];
            }
            slae(mat);

            // for(int i=0;i<n;i++){
            //     for(int j=0;j<n+1;j++){
            //         cout << mat[i][j] << " ";
            //     }
            //     cout << "\n";
            // }
            //cout << checkSoln(mat); cout << "\n";
            if(checkSoln(mat) == -1){
                cout << "inconsistent";
```

```cpp
        } else if (checkSoln(mat) == 1){
            cout << "multiple";
        } else {
            for(int i=0;i<n;i++){
                cout << mat[i][n] << " ";
            }
        }
    }
    cout << "\n";
}

}
```

4 Misc

4.1 BigInt

```cpp
#include <bits/stdc++.h>
using namespace std;
/**
 * Description: Big Integer
 * Source: https://github.com/indy256/codelibrary/
 *                  blob/master/cpp/numbertheory/bigint.cpp
 * Verification: https://oj.uz/problem/view/IOI11_parrots
 */

// base and base_digits must be consistent
constexpr int base = 1000000000;
constexpr int base_digits = 9;

struct bigint {
    // value == 0 is represented by empty z
    vector<int> z; // digits

    // sign == 1 <==> value >= 0
    // sign == -1 <==> value < 0
    int sign;

    bigint() : sign(1) {}
    bigint(long long v) { *this = v; }

    bigint &operator=(long long v) {
        sign = v < 0 ? -1 : 1; v *= sign;
        z.clear(); for (; v > 0; v = v / base)
            z.push_back((int) (v % base));
        return *this;
    }

    bigint(const string &s) { read(s); }

    bigint &operator+=(const bigint &other) {
        if (sign == other.sign) {
            for (int i = 0, carry = 0; i <
                other.z.size() || carry; ++i) {
                if (i == z.size())
                    z.push_back(0);
                z[i] += carry + (i <
                    other.z.size() ?
                    other.z[i] : 0);
```

```cpp
                carry = z[i] >= base;
                if (carry)
                    z[i] -= base;
            }
        } else if (other != 0 /* prevent infinite
            loop */) {
            *this -= -other;
        }
        return *this;
    }

    friend bigint operator+(bigint a, const bigint &b)
        { return a += b; }

    bigint &operator-=(const bigint &other) {
        if (sign == other.sign) {
            if (sign == 1 && *this >= other ||
                sign == -1 && *this <= other) {
                for (int i = 0, carry = 0; i
                    < other.z.size() ||
                    carry; ++i) {
                    z[i] -= carry + (i <
                        other.z.size() ?
                        other.z[i] : 0);
                    carry = z[i] < 0;
                    if (carry)
                        z[i] += base;
                }
                trim();
            } else {
                *this = other - *this;
                this->sign = -this->sign;
            }
        } else {
            *this += -other;
        }
        return *this;
    }

    friend bigint operator-(bigint a, const bigint &b)
        { return a -= b; }

    bigint &operator*=(int v) {
        if (v < 0) sign = -sign, v = -v;
        for (int i = 0, carry = 0; i < z.size() ||
            carry; ++i) {
            if (i == z.size())
                z.push_back(0);
            long long cur = (long long) z[i] * v
                + carry;
            carry = (int) (cur / base);
            z[i] = (int) (cur % base);
        }
        trim();
        return *this;
    }

    bigint operator*(int v) const { return
        bigint(*this) *= v; }
    friend pair<bigint, bigint> divmod(const bigint
        &a1, const bigint &b1) {
```

```cpp
        int norm = base / (b1.z.back() + 1);
        bigint a = a1.abs() * norm;
        bigint b = b1.abs() * norm;
        bigint q, r;
        q.z.resize(a.z.size());

        for (int i = (int) a.z.size() - 1; i >= 0;
            i--) {
            r *= base;
            r += a.z[i];
            int s1 = b.z.size() < r.z.size() ?
                r.z[b.z.size()] : 0;
            int s2 = b.z.size() - 1 < r.z.size()
                ? r.z[b.z.size() - 1] : 0;
            int d = (int) (((long long) s1 * base
                + s2) / b.z.back());
            r -= b * d;
            while (r < 0)
                r += b, --d;
            q.z[i] = d;
        }

        q.sign = a1.sign * b1.sign;
        r.sign = a1.sign;
        q.trim();
        r.trim();
        return {q, r / norm};
    }

    friend bigint sqrt(const bigint &a1) {
        bigint a = a1;
        while (a.z.empty() || a.z.size() % 2 == 1)
            a.z.push_back(0);

        int n = a.z.size();

        int firstDigit = (int) ::sqrt((double) a.z[n
            - 1] * base + a.z[n - 2]);
        int norm = base / (firstDigit + 1);
        a *= norm;
        a *= norm;
        while (a.z.empty() || a.z.size() % 2 == 1)
            a.z.push_back(0);

        bigint r = (long long) a.z[n - 1] * base +
            a.z[n - 2];
        firstDigit = (int) ::sqrt((double) a.z[n -
            1] * base + a.z[n - 2]);
        int q = firstDigit;
        bigint res;

        for (int j = n / 2 - 1; j >= 0; j--) {
            for (;; --q) {
                bigint r1 = (r - (res * 2 *
                    base + q) * q) * base *
                    base +
                            (j > 0 ?
                            (long
                            long)
                            a.z[2
                            *
                            j
```

```cpp
                                                       -
                                                       1]
                                                       *
                                                       base
                                                       +
                                                       a.z[2
                                                       *
                                                       j
                                                       -
                                                       2]
                                                       :
                                                       0);
                if (r1 >= 0) {
                    r = r1;
                    break;
                }
            }
            res *= base;
            res += q;

            if (j > 0) {
                int d1 = res.z.size() + 2 <
                    r.z.size() ?
                    r.z[res.z.size() + 2] :
                    0;
                int d2 = res.z.size() + 1 <
                    r.z.size() ?
                    r.z[res.z.size() + 1] :
                    0;
                int d3 = res.z.size() <
                    r.z.size() ?
                    r.z[res.z.size()] : 0;
                q = (int) (((long long) d1 *
                    base * base + (long
                    long) d2 * base + d3) /
                    (firstDigit * 2));
            }
        }
    }

    res.trim();
    return res / norm;
}

bigint operator/(const bigint &v) const { return
    divmod(*this, v).first; }

bigint operator%(const bigint &v) const { return
    divmod(*this, v).second; }

bigint &operator/=(int v) {
    if (v < 0) sign = -sign, v = -v;
    for (int i = (int) z.size() - 1, rem = 0; i
        >= 0; --i) {
        long long cur = z[i] + rem * (long
            long) base;
        z[i] = (int) (cur / v);
        rem = (int) (cur % v);
    }
    trim();
    return *this;
}
```

```cpp
bigint operator/(int v) const { return
    bigint(*this) /= v; }

int operator%(int v) const {
    if (v < 0) v = -v;
    int m = 0;
    for (int i = (int) z.size() - 1; i >= 0; --i)
        m = (int) ((z[i] + m * (long long)
            base) % v);
    return m * sign;
}

bigint &operator*=(const bigint &v) { return *this
    = *this * v; }
bigint &operator/=(const bigint &v) { return *this
    = *this / v; }

bool operator<(const bigint &v) const {
    if (sign != v.sign)
        return sign < v.sign;
    if (z.size() != v.z.size())
        return z.size() * sign < v.z.size() *
            v.sign;
    for (int i = (int) z.size() - 1; i >= 0; i--)
        if (z[i] != v.z[i])
            return z[i] * sign < v.z[i] *
                sign;
    return false;
}

bool operator>(const bigint &v) const { return v <
    *this; }
bool operator<=(const bigint &v) const { return !(v
    < *this); }
bool operator>=(const bigint &v) const { return
    !(*this < v); }

bool operator==(const bigint &v) const { return
    !(*this < v) && !(v < *this); }

bool operator!=(const bigint &v) const { return
    *this < v || v < *this; }

void trim() {
    while (!z.empty() && z.back() == 0)
        z.pop_back();
    if (z.empty()) sign = 1;
}

bool isZero() const { return z.empty(); }

friend bigint operator-(bigint v) {
    if (!v.z.empty()) v.sign = -v.sign;
    return v;
}

bigint abs() const {
    return sign == 1 ? *this : -*this;
}

long long longValue() const {
    long long res = 0;
```

```cpp
    for (int i = (int) z.size() - 1; i >= 0; i--)
        res = res * base + z[i];
    return res * sign;
}

friend bigint gcd(const bigint &a, const bigint &b)
    {
    return b.isZero() ? a : gcd(b, a % b);
}

friend bigint lcm(const bigint &a, const bigint &b)
    {
    return a / gcd(a, b) * b;
}

void read(const string &s) {
    sign = 1;
    z.clear();
    int pos = 0;
    while (pos < s.size() && (s[pos] == '-' ||
        s[pos] == '+')) {
        if (s[pos] == '-')
            sign = -sign;
        ++pos;
    }
    for (int i = (int) s.size() - 1; i >= pos; i
        -= base_digits) {
        int x = 0;
        for (int j = max(pos, i - base_digits
            + 1); j <= i; j++)
            x = x * 10 + s[j] - '0';
        z.push_back(x);
    }
    trim();
}

friend istream &operator>>(istream &stream, bigint
    &v) {
    string s; stream >> s;
    v.read(s);
    return stream;
}

friend ostream &operator<<(ostream &stream, const
    bigint &v) {
    if (v.sign == -1)
        stream << '-';
    stream << (v.z.empty() ? 0 : v.z.back());
    for (int i = (int) v.z.size() - 2; i >= 0;
        --i)
        stream << setw(base_digits) <<
            setfill('0') << v.z[i];
    return stream;
}

static vector<int> convert_base(const vector<int>
    &a, int old_digits, int new_digits) {
    vector<long long> p(max(old_digits,
        new_digits) + 1);
    p[0] = 1;
    for (int i = 1; i < p.size(); i++)
        p[i] = p[i - 1] * 10;
```

```cpp
    vector<int> res;
    long long cur = 0;
    int cur_digits = 0;
    for (int v : a) {
            cur += v * p[cur_digits];
            cur_digits += old_digits;
            while (cur_digits >= new_digits) {
                    res.push_back(int(cur %
                        p[new_digits]));
                    cur /= p[new_digits];
                    cur_digits -= new_digits;
            }
    }
    res.push_back((int) cur);
    while (!res.empty() && res.back() == 0)
            res.pop_back();
    return res;
}


typedef vector<long long> vll;

static vll karatsubaMultiply(const vll &a, const
    vll &b) {
    int n = a.size();
    vll res(n + n);
    if (n <= 32) {
            for (int i = 0; i < n; i++)
                    for (int j = 0; j < n; j++)
                            res[i + j] += a[i] *
                                b[j];
            return res;
    }

    int k = n >> 1;
    vll a1(a.begin(), a.begin() + k);
    vll a2(a.begin() + k, a.end());
    vll b1(b.begin(), b.begin() + k);
    vll b2(b.begin() + k, b.end());

    vll a1b1 = karatsubaMultiply(a1, b1);
    vll a2b2 = karatsubaMultiply(a2, b2);

    for (int i = 0; i < k; i++)
            a2[i] += a1[i];
    for (int i = 0; i < k; i++)
            b2[i] += b1[i];

    vll r = karatsubaMultiply(a2, b2);
    for (int i = 0; i < a1b1.size(); i++)
            r[i] -= a1b1[i];
    for (int i = 0; i < a2b2.size(); i++)
            r[i] -= a2b2[i];

    for (int i = 0; i < r.size(); i++)
            res[i + k] += r[i];
    for (int i = 0; i < a1b1.size(); i++)
            res[i] += a1b1[i];
    for (int i = 0; i < a2b2.size(); i++)
            res[i + n] += a2b2[i];
    return res;
}
```

```cpp
    bigint operator*(const bigint &v) const {
            vector<int> a6 = convert_base(this->z,
                base_digits, 6);
            vector<int> b6 = convert_base(v.z,
                base_digits, 6);
            vll a(a6.begin(), a6.end());
            vll b(b6.begin(), b6.end());
            while (a.size() < b.size())
                    a.push_back(0);
            while (b.size() < a.size())
                    b.push_back(0);
            while (a.size() & (a.size() - 1))
                    a.push_back(0), b.push_back(0);
            vll c = karatsubaMultiply(a, b);
            bigint res;
            res.sign = sign * v.sign;
            for (int i = 0, carry = 0; i < c.size();
                i++) {
                    long long cur = c[i] + carry;
                    res.z.push_back((int) (cur %
                        1000000));
                    carry = (int) (cur / 1000000);
            }
            res.z = convert_base(res.z, 6, base_digits);
            res.trim();
            return res;
    }
};


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

    #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif

    int t; cin >> t;
    bigint a; bigint b;
    cin >> a >> b;


}
```

4.2   Simplex

```cpp
// Simplex Method for Linear Programming
// m - number of (less than) inequalities
// n - number of variables
// C - (m+1) by (n+1) array of coefficients:
// row 0 - objective function coefficients
// row 1:m - less-than inequalities
// column 0:n-1 - inequality coefficients
// column n - inequality constants (0 for objective
    function)
// X[n] - result variables
// return value - maximum value of objective function
// (-inf for infeasible, inf for unbounded)
```

```cpp
#include <vector>
#include <cmath>

#define MAXM 400
#define MAXN 400

#define EPS 1e-9
#define INF 1.0/0.0

double A[MAXM][MAXN];
int basis[MAXM], out[MAXN];

void pivot(int m, int n, int a, int b) {
    int i, j;
    for(i = 0; i <= m; i ++) if(i != a) for(j = 0; j <= n;
        j ++) if(j != b) {
        A[i][j] -= A[a][j] * A[i][b] / A[a][b];
    }
    for(j = 0; j <= n; j ++) if(j != b) A[a][j] /= A[a][b];
    for(i = 0; i <= m; i ++) if(i != a) A[i][b] =
        -A[i][b]/A[a][b];

    A[a][b] = 1/A[a][b];

    i = basis[a];
    basis[a] = out[b];
    out[b] = i;
}

double simplex(int m, int n, double C[][MAXN], double X[])
    {
    int i, j, ii, jj;

    for(i = 1; i <= m; i ++) for(j = 0; j <= n; j ++)
            A[i][j] = C[i][j];
    for(j = 0; j <= n; j ++) A[0][j] = -C[0][j];
    for(i = 0; i <= m; i ++) basis[i] = -i;
    for(j = 0; j <= n; j ++) out[j] = j;

    for(;;) {
        for(i = ii = 1; i <= m; i ++) {
            if(A[i][n] < A[ii][n]
                    || (A[i][n] == A[ii][n] && basis[i] <
                        basis[ii])) ii = i;
        }

        if(A[ii][n] >= -EPS) {
            break;
        }

        for(j = jj = 0; j < n; j ++) {
            if(A[ii][j] < A[ii][jj]-EPS
                    || (A[ii][j] < A[ii][jj]+EPS &&
                        out[i]<out[j])) jj=j;
        }

        if(A[ii][jj] >= -EPS) return -INF;
        pivot(m,n,ii,jj);
    }

    for(;;) {
```

```cpp
        for(j = jj = 0; j < n; j ++)
            if(A[0][j] < A[0][jj]
                || (A[0][j] == A[0][jj] && out[j] <
                    out[jj])) jj = j;
        if(A[0][jj] > -EPS) break;

        for(i=1,ii=0; i <= m; i ++)
            if(A[i][jj] > EPS &&
                (!ii || A[i][n]/A[i][jj] <
                    A[ii][n]/A[ii][jj]-EPS
                    || (A[i][n]/A[i][jj] <
                        A[ii][n]/A[ii][jj]+EPS
                    && basis[i]<basis[ii]))) ii = i;

        if(A[ii][jj] <= EPS) return INF;
        pivot(m,n,ii,jj);
    }

    for(j = 0; j < n; j ++) X[j] = 0;
    for(i = 1; i <= m; i ++) if(basis[i] >= 0) X[basis[i]]
        = A[i][n];

    return A[0][n];
}
```

5 Number Theory

5.1 BinaryExponentiation

```cpp
//switch * to + for safe ll multiplication
int power(int b, int e, int m) {
    int res = 1;
    while (e > 0) {
        if (e & 1){
            res = res * b;res%=m;
        } b = b * b;b%= m;e >>= 1;
    } return res;
}
int inv(int b, int m){ return power(b,m-2,m);}
```

5.2 ExtendedEuclidean

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
//tested very little
void ext_euclid(int a, int b, int &x, int &y, int &g) {
    x = 0, y = 1, g = b;
    int m, n, q, r;
    for (int u = 1, v = 0; a != 0; g = a, a = r) {
    q = g / a, r = g % a;
    m = x- u * q, n = y- v * q;
    x = u, y = v, u = m, v = n;
    }
 }
```

```cpp
int mod_inv(int n, int m) {
    int x, y, gcd;
    ext_euclid(n, m, x, y, gcd);
    if (gcd != 1)
    return 0;
    return (x + m) % m;
}
```

6 Range Query

6.1 BIT

```cpp
#include <bits/stdc++.h>
using namespace std;
int sum(int i, vector<int> &bit){
    int res = 0; while(i>=0) res+=bit[i]; i=((i+1)&i)-1;
        return res;
}
void upd(int i, int wt, vector<int> &bit){
    while(i<bit.size()) bit[i]+=wt; i=(i+1)|i;
}
int range(int a, int b,vector<int>&bit){
    if(a == 0) return sum(b,bit); // care for indexing
    return sum(b,bit) - sum(a-1,bit);
}
```

6.2 SEGTREEBigStepper

```cpp
#include <bits/stdc++.h>
using namespace std;
template <class T> struct SegTree { // cmb(ID,b) = b
        const T ID{0};
        T cmb(T a, T b) { }
        int n; vector<T> seg;
        void init(int _n) { // upd, query also work if n =
            _n
                for (n = 1; n < _n; ) n *= 2;
                seg.assign(2*n,ID);
    }

        void pull(int p) {
        seg[p] = cmb(seg[2*p],seg[2*p+1]);
    }

        void upd(int p, T val) { // set val at position p
                seg[p += n] += val;
        for (p /= 2; p; p /= 2) pull(p);
        }
        T query(int l, int r) { // zero-indexed, inclusive
                T ra = ID, rb = ID;
                for (l += n, r += n+1; l < r; l /= 2, r /=
                    2) {
                        if (l&1) ra = cmb(ra,seg[l++]);
                        if (r&1) rb = cmb(seg[--r],rb);
                }
                return cmb(ra,rb);
```

```cpp
    }

    int bSearch(int target){
            int p = 1;
            if(seg[p] < target) return 0;
            while(p < n){
                    if(seg[2*p] < target){
                            p = 2*p+1;
                    } else {
                            p = 2*p;
                    }
            }
            return p-n+1;
    }
    // int first_at_least(int lo, int val, int ind, int
        l, int r) { // if seg stores max across range
    //      if (r < lo || val > seg[ind]) return -1;
    //      if (l == r) return l;
    //      int m = (l+r)/2;
    //      int res = first_at_least(lo,val,2*ind,l,m);
    //      if (res != -1) return res;
    //      return first_at_least(lo,val,2*ind+1,m+1,r);
    // }
};
```

6.3 SEGTREELazy

```cpp
#include <bits/stdc++.h>
struct Node{
        bool isID = false;
        int sum =0;
        Node(bool x, int s) : isID(x), sum(s){}

};

struct lNode{
        bool isID = false;
        int m=1;
        int c=0;
        lNode(bool x) : isID(x){}
};

Node idnode(true,0);
lNode lazynode(true);
template <class T, class Q> struct SegTree { // cmb(ID,b)
    = b
    const T ID{idnode}; const Q IDQ{lazynode};
    T cmb(T a, T b) {
    // if(a.isID) return b;
        // if(b.isID) return a;
            Node res(false,0);
            res.sum = (a.sum+b.sum)%mod;
            return res;
    }

        Q lazycmb(Q a, Q b){
                if(a.isID) return b;
                if(b.isID) return a;
```

```
        lNode res(false);
        res.m=(a.m*b.m)%mod;
        res.c=(a.m*b.c + a.c)%mod;
        return res;
    }

    // void cmbTQ(T a, Q b){
    //     if(b.isID) return;
    //     if(a.isID) {

    //     }
    // }
    int n; vector<T> seg; vector<Q> lazy;
    void init(int _n) { // upd, query also work if n =
        _n
        for (n = 1; n < _n; ) n *= 2;
        seg.assign(2*n,ID);
        lazy.assign(2*n,IDQ);
    }


    void printTree(){
        for(int i=1;i<2*n;i++){
            cout << seg[i].sum << " ";
        }
        cout << "\n";
    }
    void push(int node, int l, int r){
        seg[node].sum =
            ((seg[node].sum*lazy[node].m)%mod +
            (lazy[node].c*(r-l+1))%mod)%mod; //
                operation dependent
        if(l != r){
            lazy[2*node] =
                lazycmb(lazy[node],lazy[2*node]);
            lazy[2*node+1] =
                lazycmb(lazy[node],lazy[2*node+1]);
        }
        lazy[node] = IDQ;
    }
    void pull(int p) {
    seg[p] = cmb(seg[2*p],seg[2*p+1]);
}

    void upd(int l, int r, Q val){
        upd(l,r,val,0,n-1,1);
    }

    void upd(int l, int r, Q val, int start, int end,
        int node) {
        push(node,start,end);
        if(r < start || l > end) return; // maybe
            not needed

        if(l <= start && end <= r){
            lazy[node] = val;
            push(node,start,end);
            return;
        }
        int mid = (start + end)/2;
        //if(start <=l && r <= mid){
            upd(l,r,val,start,mid,2*node);
```

```
        //} else {
            upd(l,r,val,mid+1,end,2*node+1);
        //}
        pull(node);
}

    T query(int l, int r){
        return query(l,r,0,n-1,1);
    }
    T query(int l, int r, int start, int end, int node)
        { // zero-indexed, inclusive
        push(node,start,end);
        if(r < start || l > end){
            return ID;
        }
        if(l <= start && end <= r){
            return seg[node];
        } else {
            int mid = (start + end)/2;
            T x = query(l,r, start, mid,2*node);
            T y = query(l,r, mid+1, end,2*node+1);
            return cmb(x,y);
        }
    }
};
```

6.4 SEGTREERecursive

```
#include <bits/stdc++.h>
template <class T> struct SegTree { // cmb(ID,b) = b
    const T ID{0}; T cmb(T a, T b) {
    if(a == ID){
            return b;
        }
        if(b == ID){
            return a;
        }
        return min(a,b);
}

    int n; vector<T> seg;
    void init(int _n) { // upd, query also work if n =
        _n
        for (n = 1; n < _n; ) n *= 2;
        seg.assign(2*n,ID);
}

    void pull(int p) {
    seg[p] = cmb(seg[2*p],seg[2*p+1]);
}

    void upd(int p,T val) upd(p, val,0,n-1,1);
    void upd(int p, T val, int start, int end, int
        node) { // set val at position p

        if(p < start || p > end) return; // maybe
            not needed

        if(start == end){
            seg[node] = val;
            return;
```

```
        }
        int mid = (start + end)/2;
        if(start <=p && p <= mid){
            upd(p,val,start,mid,2*node);
        } else {
            upd(p,val,mid+1,end,2*node+1);
        }
        pull(node);
    }

    T query(int l, int r) query(l,r,1,0,n-1)
    T query(int l, int r, int node, int start, int end)
        { // zero-indexed, inclusive

        if(r < start || l > end){
            return ID;
        }
        if(l <= start && end <= r){
            return seg[node];
        } else {
            int mid = (start + end)/2;
            T x = query(l,r,2*node, start, mid);
            T y = query(l,r,2*node+1, mid+1, end);
            return cmb(x,y);
        }
    }
};
```

7 Syntax and Headers

7.1 CustomComparator

```
#include <bits/stdc++.h>
using namespace std;
struct cc{
    bool operator()(const int &a, const int &b)
        const{return b<a;}
};
set<int,cc> S;
```

7.2 CustomHash

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define f first
#define s second
#define pl pair<ll, ll>
struct pair_hash {
    size_t operator()(const pair<pl, ll>& p) const {
        auto hash1 = hash<ll>{}(p.f.f);
        auto hash2 = hash<ll>{}(p.f.s);
        auto hash3 = hash<ll>{}(p.s);
        return hash1 ^ (hash2 << 1) ^ (hash3 << 2);
```

```
        }
};
unordered_map<pair<pl,ll>, ll, pair_hash> dp;
```

7.3   StringBitsetOperations

```
#include <bits/stdc++.h>
using namespace std;
```

8   Trees

8.1   LCA

```
#include <bits/stdc++.h>
#define pb push_back
using namespace std;

int n; int q;
int par[200005][21];
int depth[200005];
vector<int> adj[200005];

void buildArr(int node, int p){
    par[node][0] = p;
    for(int i=1;i<20;i++){
        if(par[node][i-1] != -1){
            par[node][i] = par[par[node][i-1]][i-1];
```

```
        }
    }
    if(p == -1) depth[node] = 0;
    else depth[node] = depth[p] + 1;

    for(auto x : adj[node]){
        if(x == p) continue;
        buildArr(x,node);
    }
}


int bigStepper(int node, int k){
    int x = 0;
    for(int i=0;i<20;i++){
        if(k%2==1) node = par[node][i];
        k /= 2;
    }
    return node;
}


int lca(int a, int b){
    if (depth[a] > depth[b]) swap(a,b);

    b = bigStepper(b,depth[b] - depth[a]);
    //cout << b;
    if(a == b) return a;
    for(int i=19;i>=0;i--){
        if(par[a][i] != par[b][i]){
            a = par[a][i];
            b = par[b][i];
        }
```

```
    }
    return par[a][0];
}
signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif
    cin >> n >> q;

    for(int i=0;i<=n;i++){
        for(int j=0;j<20;j++){
            par[i][j] = -1;
        }
    }

    for(int i=0;i<n-1;i++){
        int a; int b; cin >> a >> b;
        adj[a].pb(b);
        adj[b].pb(a);
    }

    buildArr(1,-1);
    for(int i =0;i<q;i++){
        int a; int b; cin >> a >>b;
        cout << depth[a] + depth[b] - 2*depth[lca(a,b)] <<
            "\n";
    }
}
```