Notebook - Maratona de Programação

isasouza

Contents

1 All

1.1 Bfs

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

using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define pb push_back
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);

const int INF = 0x3f3f3f3f3f3f;
const long double PI = acos(-1);
int maximo = 1e5+10;
vector <vector<int>> adj(maximo);
vector <int> bfs(int s, int N)
{
    vector<int> dist(N+1, INF);
    queue <int> q;

    dist[s] = 0;
    q.push(s);
    while(!q.empty())
    {
        auto u = q.front();
        q.pop();

        for(auto v : adj[u])
        {
            if(dist[v] > dist[u] + 1)
            {
                dist[v] = dist[u] + 1;
                q.push(v);
            }
        }
    }
    return dist;
}
int32_t main()
{   sws
    int n, m;
    cin >> n >> m;
    for(int i = 0; i < m; i++)
    {
        int x,y;
        cin >> x >> y;
        adj[x].pb(y);
    }
    int s,t;
    cin >> s >> t;
    vector <int> d;
    d = bfs(s, n);
    int ans = d[t];
    if(ans != INF) cout << "YES\n";
    else cout << "NO\n";
    return 0;
}
```

1.2 Binary Lifting

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

using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define pb push_back
#define endl "\n"
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);

const int INF = 0x3f3f3f3f3f3f;
const long double PI = acos(-1);
const int MAX = 2e5;
const int LOG = 30;

vector<int> adj[MAX];
int up[MAX][LOG], parent[MAX];

void process(int n)
{
    for(int v = 1; v <= n; v++)
    {
        up[v][0] = parent[v];
        for(int i = 1; i < LOG; i++)
        {
            up[v][i] = up[up[v][i-1]][i-1];
        }
    }
}

int jump(int n, int k)
{
    for(int i = 0 ; i < LOG; i++)
    {
        if(k & ( 1 << i))
        {
            n = up[n][i];
        }
    }
    if(n == 0) return -1;
    return n;
}
int32_t main()
{   sws
    int n,q;
    cin >> n >> q;

    parent[1] = 0;
    for(int i = 1; i <= n; i++)
    {
        int x;
        cin >> x;
        parent[i] = x;

        adj[i].pb(x);
        adj[x].pb(i);
    }

    process(n);
    for(int i = 1; i <= q; i++)
    {
        int a,k;
        cin >> a >> k;
        cout << jump(a,k) << '\n';
    }
    return 0;
}
```

1.3 Convex Hull

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

using namespace std;
#define int long long
typedef int cod;

struct point
{
    cod x,y;
    point(cod x = 0, cod y = 0): x(x), y(y)
    {}

    double modulo()
    {
        return sqrt(x*x + y*y);
    }

    point operator+(point o)
    {
        return point(x+o.x, y+o.y);
    }
    point operator-(point o)
    {
        return point(x - o.x , y - o.y);
    }
```

```
    point operator*(cod t)
    {
        return point(x*t, y*t);
    }
    point operator/(cod t)
    {
        return point(x/t, y/t);
    }

    cod operator*(point o)
    {
        return x*o.x + y*o.y;
    }
    cod operator^(point o)
    {
        return x*o.y - y * o.x;
    }
    bool operator<(point o)
    {
        if( x != o.x) return x < o.x;
        return y < o.y;
    }

};

int ccw(point p1, point p2, point p3)
{
    cod cross = (p2-p1) ^ (p3-p1);
    if(cross == 0) return 0;
    else if(cross < 0) return -1;
    else return 1;
}

vector <point> convex_hull(vector<point> p)
{
    sort(p.begin(), p.end());
    vector<point> L,U;

    //Lower
    for(auto pp : p)
    {
        while(L.size() >= 2 and ccw(L[L.size() - 2], L
    .back(), pp) == -1)
        {
            // é -1 pq eu ãno quero excluir os
    colineares
            L.pop_back();
        }
        L.push_back(pp);
    }

    reverse(p.begin(), p.end());

    //Upper
    for(auto pp : p)
    {
        while(U.size() >= 2 and ccw(U[U.size()-2], U .
    back(), pp) == -1)
        {
            U.pop_back();
        }
        U.push_back(pp);
    }

    L.pop_back();
    L.insert(L.end(), U.begin(), U.end()-1);
    return L;
}

cod area(vector<point> v)
{
    int ans = 0;
    int aux = (int)v.size();
    for(int i = 2; i < aux; i++)
    {
        ans += ((v[i] - v[0])^(v[i-1] - v[0]))/2;
    }
    ans = abs(ans);
    return ans;
}

int bound(point p1 , point p2)
{
    return __gcd(abs(p1.x-p2.x), abs(p1.y-p2.y));
}
```

```
}
//teorema de pick [pontos = A - (bound+points)/2 + 1]

int32_t main()
{

    int n;
    cin >> n;

    vector<point> v(n);
    for(int i = 0; i < n; i++)
    {
        cin >> v[i].x >> v[i].y;
    }

    vector <point> ch = convex_hull(v);

    cout << ch.size() <<  '\n';
    for(auto p : ch) cout << p.x << " " << p.y << "\n"
    ;

    return 0;
}
```

1.4 Dfs Tree

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

using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define pb push_back
#define endl "\n"
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);
#define ff first
#define ss second

const int INF = 0x3f3f3f3f3f3f;
const long double PI = acos(-1);
const int MAX = 4e5 + 10;
int n,m;
vector<pair<int,int>> graph[MAX];
vector<pair<int,int>> edge(MAX);
bool bridge[MAX];
bool vis[MAX];
int tin[MAX], low[MAX], comp[MAX],st[MAX];
int componentes = 0, temp = 0, aux;
vector<int> vert_com[MAX];

void dfs(int u, int p)
{
    vis[u] = true;
    temp++;
    tin[u] = low[u]= temp;
    aux++;
    st[aux] = u;
    for(auto [v, idx] : graph[u])
    {
        if(v == p) continue;
        if(vis[v])
        {
            low[u] = min(low[u], tin[v]);
            continue;
        }
        dfs(v,u);
        low[v] = min(low[v], low[u]);
        if(low[v] > tin[u])
        {
            bridge[idx] = true;
        }
    }
    if(low[u] == tin[u])
    {
        componentes++;
        int vert;
        do
        {
            cout << "u: " <<  u << endl;
```

```
            vert = st[aux];
            aux--;
            comp[vert] = componentes;
            vert_com[componentes].pb(vert);
        } while (vert != u);
    }
}
bool vis2[MAX];
bool used[MAX];
void dfs2(int u, int p)
{
    vis2[u] = true;
    for(auto [v, idx] : graph[u])
    {
        if(v == p) continue;
        if(!used[idx])
        {
            if(bridge[idx]) edge[idx] = {v,u};
            else edge[idx] = {u,v};
            used[idx] = true;
        }
        if(vis[v]) continue;
        dfs2(v,u);
    }
}


int32_t main()
{   sws
    cin >> n >> m;
    for(int i = 1; i <= m; i++)
    {
        int a,b;
        cin >> a >> b;
        graph[a].push_back({b,i});
        graph[b].push_back({a,i});
    }
    dfs(1,0);

    int mx = 0,  ini = 0;
    for(int i = 1; i <= componentes; i++)
    {
        if(vert_com[i].size() > mx)
        {
            ini = i;
            mx = vert_com[i].size();
        }
    }
    for(auto x : vert_com[ini])
    {
        cout << x << " ";
    }
    cout << endl;
    //cout << "ni: " << ini << ' ' << mx << endl;

    dfs2(vert_com[ini][0], vert_com[ini][0]);

    for(int i = 1; i <= m; i++)
    {
        int a = edge[i].ff;
        int b = edge[i].ss;
        cout << a << " " << b << endl;
    }
    return 0;
}
```

1.5 Dinic

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

using namespace std;
#define ll long long
#define pb push_back
const ll LLINF = INT64_MAX;

const int N = 501;

struct Dinic {
    struct Edge{
        int from, to; ll flow, cap;
    };
    vector<Edge> edge;

    vector<int> g[N];
    int ne = 0;
    int lvl[N], vis[N], pass;
    int qu[N], px[N], qt;

    ll run(int s, int sink, ll minE) {
        if(s == sink) return minE;

        ll ans = 0;

        for(; px[s] < (int)g[s].size(); px[s]++) {
            int e = g[s][ px[s] ];
            auto &v = edge[e], &rev = edge[e^1];
            if(lvl[v.to] != lvl[s]+1 || v.flow >= v.
    cap)
                continue;           // v.cap - v.flow
    < lim
            ll tmp = run(v.to, sink,min(minE, v.cap-v.
    flow));
            v.flow += tmp, rev.flow -= tmp;
            ans += tmp, minE -= tmp;
            if(minE == 0) break;
        }
        return ans;
    }
    bool bfs(int source, int sink) {
        qt = 0;
        qu[qt++] = source;
        lvl[source] = 1;
        vis[source] = ++pass;
        for(int i = 0; i < qt; i++) {
            int u = qu[i];
            px[u] = 0;
            if(u == sink) return true;
            for(auto& ed : g[u]) {
                auto v = edge[ed];
                if(v.flow >= v.cap || vis[v.to] ==
    pass)
                    continue; // v.cap - v.flow < lim
                vis[v.to] = pass;
                lvl[v.to] = lvl[u]+1;
                qu[qt++] = v.to;
            }
        }
        return false;
    }
    ll flow(int source, int sink) {
        reset_flow();
        ll ans = 0;
        //for(lim = (1LL << 62); lim >= 1; lim /= 2)
        while(bfs(source, sink))
            ans += run(source, sink, LLINF);
        return ans;
    }
    void addEdge(int u, int v, ll c, ll rc) {
        Edge e = {u, v, 0, c};
        edge.pb(e);
        g[u].push_back(ne++);

        e = {v, u, 0, rc};
        edge.pb(e);
        g[v].push_back(ne++);
    }
    void reset_flow() {
        for(int i = 0; i < ne; i++)
            edge[i].flow = 0;
        memset(lvl, 0, sizeof(lvl));
        memset(vis, 0, sizeof(vis));
        memset(qu, 0, sizeof(qu));
        memset(px, 0, sizeof(px));
        qt = 0; pass = 0;
    }
    vector<pair<int, int>> cut() {
        vector<pair<int, int>> cuts;
        for (auto [from, to, flow, cap]: edge) {
            if (flow == cap and vis[from] == pass and
    vis[to] < pass and cap>0) {
                cuts.pb({from, to});
            }
        }
        return cuts;
    }
};

int main()
```

```
{
    Dinic dinic;
    ll n, m;
    cin >> n >> m;
    for(int i = 0; i < m; i++)
    {
        ll a,b,c;
        cin >> a >> b >> c;
        dinic.addEdge(a,b,c,0);
    }
    ll ans = dinic.flow(1,n);
    cout << ans << '\n';
    return 0;
}
```

1.6 Djikstra

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

using namespace std;

const int maxn = 1010;
const int INF = 0x3f3f3f3f;
#define pii pair<int,int>
vector<vector<pii>> adj(maxn);

vector<int> djikstra(int ini)
{
    vector<int> distance(maxn, INF);
    vector<bool> vis(maxn, false);
    priority_queue<pii, vector<pii>, greater<pii>> pq;

    distance[ini] = 0;
    pq.push({0,ini});

    while(!pq.empty())
    {
        int u = pq.top().second;
        pq.pop();

        if(vis[u]) continue;

        for(auto aux : adj[u])
        {
            int v = aux.first;
            int custo = aux.second;

            if(distance[v] > distance[u] + custo)
            {
                distance[v] = distance[u] + custo;
                pq.push({distance[v], v});
            }
        }
    }

    return distance;
}
int main()
{
    int n, m;
    cin >> n >> m;
    for(int i = 0; i < m; i++)
    {
        int a,b,c;
        cin >> a >> b >> c;
        adj[a].push_back({b,c});
        adj[b].push_back({a,c});
    }

    vector <int> d = djikstra(1);

    for(int i = 1; i <= n; i++)
    {
        cout << "distancia ate " << i << " e: " << d[i
    ] << endl;
    }
    return 0;
}
```

1.7 Dsu

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

// Complexidade
// build : O(N)
// find : O(logN)
class DSU
{
    public:
    vector<int> parent, sz;
    void make(int v)
    {
        parent[v] = v;
        sz[v] = 1;
    }

    int find(int v)
    {
        if (v == parent[v]) return v;
        return parent[v] = find(parent[v]);
    }

    void union_(int a, int b)
    {
        a = find(a), b = find(b);

        if(sz[b]>sz[a]) swap(a,b);
        if (a != b)
        {
            sz[a] += sz[b];
            parent[b] = a;
        }
    }

    bool same(int a, int b)
    {
        a = find(a), b = find(b);
        return a == b;
    }

    DSU(int n): parent(n+1), sz(n+1)
    {
        for(int i=1; i<=n; i++) make(i);
    }
};

void solve()
{
    int n;
    cin >> n;
    DSU dsu(n);
}

int main()
{
    int t = 1;
    while(t--)
    {
        solve();
    }
    return 0;
}
```

1.8 Ford Fulkerson And Edmonds Karp

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

using namespace std;
#define int long long
#define pb push_back

// Description:
// Obtains the maximum possible flow rate given a
    network. A network is a graph with a single source
     vertex and a single sink vertex in which each
    edge has a capacity

// Complexity:
// O(V * E^2) where V is the number of vertex and E is
     the number of edges
const int MAXN = 501;
const int MAXE = 1001;
const int INF = INT64_MAX;

// represents the capacities of the edges
int capacity[MAXN][MAXE];
```

```
// represents the graph and it may contain negative
    edges
vector<int> adj[MAXN];
int n, e;


int bfs(int s, int t, vector<int>& parent) {
    fill(parent.begin(), parent.end(), -1);
    parent[s] = -2;
    queue<pair<int, int>> q;
    q.push({s, INF});

    while (!q.empty()) {
        int cur = q.front().first;
        int flow = q.front().second;
        q.pop();

        for (int next : adj[cur])
        {
            //cout << "cur next " << cur << ' ' <<
    next <<  ' ' << parent[next] << ' ' << capacity[
    cur][next] << endl;
            if (parent[next] == -1 && capacity[cur][
    next])
            {
                parent[next] = cur;
                int new_flow = min(flow, capacity[cur
    ][next]);
                if (next == t)
                {
                    //cout << new_flow << endl;
                    return new_flow;
                }
                q.push({next, new_flow});
            }
        }
    }

    return 0;
}

int maxflow(int s, int t) {
    int flow = 0;
    vector<int> parent(n+1);
    int new_flow;

    while (new_flow = bfs(s, t, parent)) {
        flow += new_flow;
        int cur = t;
        while (cur != s) {
            int prev = parent[cur];
            capacity[prev][cur] -= new_flow;
            capacity[cur][prev] += new_flow;
            cur = prev;
        }
    }

    return flow;
}

int32_t main()
{
    cin>>n>>e;
    int s = 1, t = n;
    //cin>>s>>t;

    for(int i = 0; i < e; i++)
    {
        int from, to, cap;
        cin>>from>>to>>cap;

        capacity[from][to] += cap;
        adj[from].push_back(to);
        //adding the negative edges
        adj[to].push_back(from);
    }

    // for(int i = 1; i <= n; i++)
    // {   cout << i << " : ";
    //     for(auto x : graph[i]) cout << x << ' ';
    //     cout << endl;
    // }

    int maxFlow = maxflow(s, t);
    cout<<maxFlow<<endl;

    return 0;
}
```

1.9 Knapsack

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

using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define pb push_back
#define endl "\n"
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);

const int INF = 0x3f3f3f3f3f3f;
const long double PI = acos(-1);
const int MAX = 1010;
const int MOD = 1e9 + 7;

int dp[MAX][MAX], custo[MAX], valor[MAX];
int n, s;

int knap(int i, int c)
{
    if(c < 0) return -INF;
    if(i == n) return 0;

    if(dp[i][c] != -1) return dp[i][c];

    return dp[i][c] = max(knap(i+1, c - custo[i]) +
    valor[i], knap(i+1, c));
}

vector <int> path;
void recover(int i , int c)
{
    if( i == n) return;

    int pega = knap(i + 1, c - custo[i]) + valor[i];
    int npega = knap(i+1, c);
    if(pega >= npega)
    {
        path.pb(i);
        recover(i+1, c - custo[i]);
    }
    else
    {
        recover(i+1,c);
    }
}
int32_t main()
{   sws

    memset(dp,-1,sizeof(dp));
    cin >> n >> s;
    for(int i = 0; i < n; i++)
    {
        cin >> custo[i] >> valor[i];
    }

    recover(0,s);
    int peso = 0;
    for(int x : path)
    {
        peso += custo[x];
        cout << "Pega o valor " << valor[x] << " com
    peso " << custo[i] << "com idx " << x << "\n";
    }
    cout << "Pega " << path.size() << " itens com o
    peso total de " << peso << " e valor total de " <<
     knap(0,s) << "\n";
    return 0;
}
```

1.10 Kosaraju

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

using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define pb push_back
#define endl "\n"
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);
#define ff first
#define ss second
 const int INF = 0x3f3f3f3f3f;
const long double PI = acos(-1);
const int MAX = 2e5;

int n,q,d;
int componente[MAX];
vector<int> adj[MAX];
vector<int> adj2[MAX];
vector<int> saida;
int vis[MAX];

void dfs(int u)
{
    vis[u] = 1;
    for(auto v : adj[u])
    {
        if(!vis[v])
        {
            dfs(v);
        }
    }
    saida.pb(u);
}

void dfs2(int u, int c)
{
    vis[u] = 2;
    componente[u] = c;
    for(auto v : adj2[u])
    {
        if(vis[v] == 1) dfs2(v, c);
    }
}


void solve()
{
    cin >> n  >> m;

    for(int i = 0; i < m; i++)
    {
        int x,y;
        cin >> x >> y;
        adj[x].pb(y);
        adj2[y].pb(x);
    }



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

    int c = 0;
    for(int i = saida.size() - 1; i >= 0; i--)
    {
        if(vis[saida[i]] == 1)
        {
            c++;
            dfs2(saida[i], c);
        }
    }
    return;
}
int32_t main()
{   sws

    int t = 1;
    while(t--) solve();
    return 0;
}
```

1.11 Lazy

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

using namespace std;

#define ff first
#define ss second
#define int long long
#define pb push_back
#define sws ios_base::sync_with_stdio(false);cin.tie(
    NULL);cout.tie(NULL);


const int MAX = 2e5 + 2;
const int INF = 0x3f3f3f3f;


vector <int> lazy(4*MAX, 0);
int tree[4*MAX], v[MAX];
int N;

int merge(int a, int b)
{
    return a + b;
}

void build(int l, int r, int id)
{
    if(l==r)
    {
        tree[id] = v[l];
        return;
    }
    int mid = (l+r)/2;
    build(l, mid, 2*id);
    build(mid+1, r, 2*id+1);

    tree[id] = merge(tree[2*id], tree[2*id+1]);
}

void prop(int l, int r, int id)
{
    if(lazy[id]!=0)
    {
        tree[id] += (r-l+1)*lazy[id];
        if(l!=r)
        {
            lazy[2*id] += lazy[id] ;
            lazy[2*id+1] += lazy[id];
        }
        lazy[id] = 0;
    }
}

void update(int A, int B, int x, int l=0, int r=N-1,
    int id=1)
{
    prop(l, r, id);
    // º1 caso
    if(B<l or r<A) return;
    // º2 caso
    if(A<=l and r<=B)
    {
        lazy[id] += x;
        prop(l, r, id);
        return;
    }
    // º3 caso
    int mid = (l+r)/2;

    update(A, B, x, l, mid, 2*id);
    update(A, B, x, mid+1, r, 2*id+1);
```

```
    tree[id] = merge(tree[2*id], tree[2*id+1]);
}

int query(int A, int B, int l=0, int r=N-1, int id=1)
{
    prop(l, r, id);
    // º1 caso
    if(B<l or r<A) return 0;
    // º2 caso
    if(A<=l and r<=B) return tree[id];
    // º3 caso
    int mid = (l+r)/2;

    return merge(query(A, B, l, mid, 2*id),query(A, B,
     mid+1, r, 2*id+1));
}


int32_t main()
{
    sws;

    int Q, op, a, b, idx, x;
    cin >> N >> Q;
    for(int i=0;i<N;i++)
    {
        cin >> v[i];
    }

    build(0,N-1,1);

    for(int i=0;i<Q;i++)
    {
        cin >> op;
        if(op==1)
        { // update
            cin >> a >> b >> x;
            a--;b--;
            update(a, b, x);


        }
        else
        { // query
            cin >> idx ;
            idx--; // indice indexado em 0
            cout << query(idx, idx) << endl;

        }
    }

    return 0;
}
```

1.12 Lca

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

using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define pb push_back
#define endl "\n"
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);

const int INF = 0x3f3f3f3f3f;
const long double PI = acos(-1);
const int MAX = 2e5 + 5;
const int LOG = 30;

int up[MAX][LOG], parent[MAX], depth[MAX];

void process(int n)
{
    for(int v = 1; v <= n; v++)
    {
        if(v != 1) depth[v] = depth[parent[v]] + 1;
        up[v][0] = parent[v];
        for(int i = 1; i < LOG; i++)
        {
            up[v][i] = up[up[v][i-1]][i-1];
        }
    }
}

int jump(int n, int k)
{
    for(int i = 0 ; i < LOG; i++)
    {
        if(k & ( 1 << i))
        {
            n = up[n][i];
        }
    }
    if(n == 0) return -1;
    return n;
}
int lca(int node1 , int node2)
{
    if(depth[node2] > depth[node1]) swap(node1,node2);
    int k = depth[node1] - depth[node2];
    node1 = jump(node1, k);

    if(node1 == node2) return node1;

    for(int i = LOG - 1; i >= 0; i--)
    {
        if(up[node1][i] != up[node2][i])
        {
            node1 = up[node1][i];
            node2 = up[node2][i];
        }
    }

    return up[node1][0];
}
int32_t main()
{   sws
    int n,q;
    cin >> n >> q;

    parent[1] = 0;
    for(int i = 2; i <= n; i++)
    {
        int x;
        cin >> x;
        parent[i] = x;

    }

    process(n);

    while(q--)
    {
        int a,b;
        cin >> a >> b;
        int ans = lca(a,b);
        cout << ans << '\n';

    }
    return 0;
}
```

1.13 Pick Theorem

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

using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define pb push_back
#define endl "\n"
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);

const int INF = 0x3f3f3f3f3f;
```

```
const long double PI = acos(-1);
typedef int cod;
struct point
{
    cod x,y;
    point(cod x = 0, cod y = 0): x(x), y(y)
    {}

    double modulo()
    {
        return sqrt(x*x + y*y);
    }

    point operator+(point o)
    {
        return point(x+o.x, y+o.y);
    }
    point operator-(point o)
    {
        return point(x - o.x , y - o.y);
    }

    cod operator^(point o)
    {
        return x*o.y - y * o.x;
    }

};

cod area(vector<point> v)
{
    int ans = 0;
    int aux = (int)v.size();
    for(int i = 2; i < aux; i++)
    {
        ans += ((v[i] - v[0])^(v[i-1] - v[0]));
    }
    ans = abs(ans);
    return ans;
}

int bound(point p1 , point p2)
{
    return __gcd(abs(p1.x-p2.x), abs(p1.y-p2.y));
}
int32_t main()
{   sws

    int n;
    cin >> n;

    vector<point> v;
    for(int i = 0; i < n; i++)
    {
        point p1;
        cin >> p1.x >> p1.y;
        v.pb(p1);
    }
    v.pb(v[0]);
    int b = 0;
    for(int i = 0; i < n; i++)
    {
        b += bound(v[i], v[i+1]);
    }

    cod A = area(v)/2;
    ///teorema de pick
    int ans = A - (b/2) + 1;
    //cout << A << '\n';
    cout << ans << " " << b << '\n';

    return 0;
}
```

1.14 Psum2d

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

using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define pb push_back
#define endl "\n"
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);

const int INF = 0x3f3f3f3f3f;
const long double PI = acos(-1);
const int MAXN = 1000 + 5;
int bosque[MAXN][MAXN];
int psum[MAXN][MAXN];

int32_t main()
{   sws
    int n, q;
    cin >> n >> q;
    /*
    for(int i = 0; i <= n; i++)
    {
        psum[i][0] = 0;
        psum[0][i] = 0;
    }
    */
    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= n; j++)
        {
            char c;
            cin >> c;
            if(c == '*')
            {
                bosque[i][j] = 1;
            }
        }
    }

    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= n; j++)
        {
            psum[i][j] = bosque[i][j] + psum[i-1][j] +
     psum[i][j-1] - psum[i-1][j-1];
            //cout << psum[i][j] << " ";
        }
        //cout << "\n";
    }

    for(int i = 0; i < q; i++)
    {
        int x1,y1,x2,y2;
        cin >> y1 >> x1 >> y2 >> x2;
        int ans = psum[y2][x2] - psum[y2][x1-1] - psum
    [y1-1][x2] + psum[y1-1][x1-1];
        cout << ans << "\n";
    }
    return 0;
}
```

1.15 Seg

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

using namespace std;
#define int long long
const int INF = 0x3f3f3f3f;
const int MAX = 2e5 + 10;
int v[MAX + 2];
int tree[4*MAX + 2];

int merge(int a, int b)
{
    return min(a,b);
}
void update(int idx, int x, int id , int il,int ir)
{
    if(il == ir)
    {
        tree[id] = x;
        //cout << "idx: " << idx << " update: " << x
    << endl;
        return;
    }
```

```
    int mid = (il + ir) / 2;

    if(mid < idx)
    {
        update(idx, x, 2*id+1, mid+1, ir);
    }
    else
    {
        update(idx, x, 2*id, il, mid);
    }

    tree[id] = merge(tree[2*id] , tree[2*id + 1]);
    //cout << "id: " << id << "update: " << tree[id]
    << endl;
    return;


}
void build(int id, int il, int ir)
{
    if(il == ir)
    {
        //Se os dois forem iguais, é ele msm
        tree[id] = v[il];
        return;
    }

    //Calcular o meio dos dois e construir a árvore
    dos filhos
    int mid = (il + ir) / 2;

    build(2*id, il, mid);
    build(2*id + 1, mid + 1, ir);

    // ans vai ser o minimo desses dois filhos

    tree[id] = merge(tree[2*id], tree[2*id + 1]);
    return;

}

int query(int l, int r, int id, int il, int ir)
{
    //se tiver no range, retorna ele
    if(il >= l && ir <= r)
    {
        return tree[id];
    }

    if(l > ir || r < il) return 0;

    int mid = (il+ir) / 2;
    int left = query(l,r,2*id, il, mid);
    int right = query(l,r,2*id+1,mid + 1, ir);

    return merge(left ,right);

}
int32_t main()
{
    int N,Q;
    cin >> N >> Q;
    //cout << N << ' ' << Q << endl;
    ;
    for(int i = 0; i < N; i++)
    {
        cin >> v[i];
    }

    build(1, 0, N-1);



    for(int i = 0; i < Q; i++)
    {
        int y, l, r;
        cin >> y >> l >> r;
        l--;r--;
        if(y == 1)
        {
            int k = l;
            int x = r;
            v[k] = x;
            update(k,x,1,0,N-1);
        }
        else if(y == 2)
        {
            int ans = query(l,r,1,0,N-1);
            cout << ans << '\n';
        }
    }


    return 0;
}
```

1.16 Template

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

using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define vvi vector<vector<int>>
#define pb push_back
#define all(x) x.begin(), x.end()
#define endl "\n"
#define ff first
#define ss second
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);

const int INF = 0x3f3f3f3f3f;
const long double PI = acos(-1);
const int MAX = 2e5 + 5;
const int MOD = 1e9 + 7;


void solve()
{

    return;
}
int32_t main()
{   sws

    int t;
    t = 1;
    // cin >>> t;
    while(t--)
    {
        solve();
    }
    return 0;
}
```

1.17 Test

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

#define int long long
using namespace std;
#define ff first
#define ss second

bool aprov(string ans1, string ans2)
{
    if((int)ans1.size() == (int)ans2.size())
    {
        bool eh = true;
        for(int k = 0; k < (int)ans1.size(); k++)
        {
            if(ans1[k] != ans2[k])
            {
                eh = false;
            }
        }
        if(eh)
        {
            cout << "Aprovado\n";
            return true;
        }
```

```
    }
    return false;
}
int32_t main()
{
    int n;
    cin >> n;
    vector <pair<string,string>> v(n);

    for(int i = 0 ; i < n; i++)
    {
        string s1,s2;
        cin >> s1 >> s2;
        v[i].first = s1;
        v[i].second = s2;
    }
    bool resp = false;
    string ans1 = "";
    string ans2 = "";
    int cnt = 0;
    vector <bool> vis(n, false);
    for(int i1 = 0; i1 < n; i1++)
    {
        cnt = 1;
        if(cnt > n)break;
        ans1 = v[i1].first;
        ans2 = v[i1].second;
        bool eh = aprov(ans1, ans2);
        if(eh)
        {
            cout << cnt << '\n';
            cout << i1 +1 << "\n";
            return 0;
        }
        for(int i2 = 0; i2 < n; i2++)
        {
            for(int aux = 0; aux < n; aux++) vis[aux]
    = false;
            vis[i1] = true;
            cnt = 2;
            if(cnt > n) break;
            if(vis[i2]) continue;
            ans1 = v[i1].first + v[i2].first;
            ans2 = v[i1].second + v[i2].second;
            bool eh = aprov(ans1, ans2);
            if(eh)
            {
                cout << cnt << '\n';
                cout << i1 +1<< " " << i2+1 << "\n";
                return 0;
            }
            for(int i3 = 0; i3 < n; i3++)
            {
                for(int aux = 0; aux < n; aux++) vis[
    aux] = false;
                vis[i1] = vis[i2] = true;
                cnt = 3;
                if(cnt > n) break;
                if(vis[i3]) continue;
                ans1 = v[i1].first + v[i2].first + v[
    i3].ff ;
                ans2 = v[i1].second + v[i2].second + v
    [i3].ss;
                bool eh = aprov(ans1, ans2);
                if(eh)
                {
                    cout << cnt << '\n';
                    cout << i1+1 << " " << i2+1 << " "
     << i3 +1<< "\n";
                    return 0;
                }
                for(int i4 = 0; i4 < n; i4++)
                {
                    for(int aux = 0; aux < n; aux++)
    vis[aux] = false;
                    vis[i1] = vis[i2] = vis[i3] = true
    ;
                    cnt = 4;
                    if(cnt > n) break;
                    if(vis[i4])continue;
                    ans1 = v[i1].first + v[i2].first +
     v[i3].ff + v[i4].ff;
                    ans2 = v[i1].second + v[i2].second
     + v[i3].ss + v[i4].ss;
                    bool eh = aprov(ans1, ans2);
                    if(eh)
                    {
                        cout << cnt << '\n';
                        cout << i1+1 << " " << i2 +1
    << " " << i3+1 << " " << i4+1 << "\n";
                        return 0;
                    }
                    for(int i5 = 0; i5 < n; i5++)
                    {
                        for(int aux = 0; aux < n; aux
    ++) vis[aux] = false;
                        vis[i1] = vis[i2] = vis[i3] =
    vis[i4] = true;
                        cnt = 5;
                        if(cnt > n) break;
                        if(vis[i5] ) continue;
                        ans1 = v[i1].first + v[i2].
    first + v[i3].ff + v[i4].ff + v[i5].ff;
                        ans2 = v[i1].second + v[i2].
    second + v[i3].ss + v[i4].ss + v[i5].ss;
                        bool eh = aprov(ans1, ans2);
                        if(eh)
                        {
                            cout << cnt << '\n';
                            cout << i1+1 << " " << i2
    +1 << " " << i3+1 << " " << i4 +1 << " " << i5+1
    << "\n";
                            return 0;
                        }
                        for(int i6 = 0; i6 < n; i6++)
                        {
                            for(int aux = 0; aux < n;
    aux++) vis[aux] = false;
                            vis[i1] = vis[i2] = vis[i3
    ] = vis[i4] = vis[i5] = true;
                            cnt = 6;
                            if(cnt > n ) break;
                            if(vis[i6])continue;
                            ans1 = v[i1].first + v[i2
    ].first + v[i3].ff + v[i4].ff + v[i5].ff + v[i6].
    ff;
                            ans2 = v[i1].second + v[i2
    ].second + v[i3].ss + v[i4].ss + v[i5].ss + v[i6].
    ss;
                            bool eh = aprov(ans1, ans2
    );
                            if(eh)
                            {
                                cout << cnt << '\n';
                                cout << i1+1 << " " <<
     i2+1 << " " << i3 +1 << " " << i4+1 << " "<< i5+1
      << " " << i6+1 << "\n";
                                return 0;
                            }
                            for(int i7 = 0; i7 < n; i7
    ++)
                            {
                                for(int aux = 0; aux <
     n; aux++) vis[aux] = false;
                                vis[i1] = vis[i2] =
    vis[i3] = vis[i4] = vis[i5] = vis[i6] = true;
                                cnt = 7;
                                if(cnt > n) break;
                                if(vis[i7])continue;
                                ans1 = v[i1].first + v
    [i2].first + v[i3].ff + v[i4].ff + v[i5].ff + v[i6
    ].ff + v[i7].ff;
                                ans2 = v[i1].second +
    v[i2].second + v[i3].ss + v[i4].ss + v[i5].ss + v[
    i6].ss + v[i7].ss;
                                bool eh = aprov(ans1,
    ans2);
                                if(eh)
                                {
                                    cout << cnt << '\n
    ';
                                    cout << i1 +1 << "
     " << i2+1 << " " << i3 +1<< " " << i4+1 << " "
    << i5 +1 << " " << i6 +1 << " " << i7 +1<< "\n";
                                    return 0;
                                }
                                for(int i8 = 0; i8 <
    8; i8++)
```

```
                                {
                                    for(int aux = 0;
    aux < n; aux++) vis[aux] = false;
                                    vis[i1] = vis[i2]
    = vis[i3] = vis[i4] = vis[i5] = vis[i6] = vis[i7]
    = true;
                                    cnt = 8;
                                    if(cnt > n) break;
                                    if(vis[i8])
    continue;
                                    ans1 = v[i1].first
     + v[i2].first + v[i3].ff + v[i4].ff + v[i5].ff +
    v[i6].ff + v[i7].ff + v[i8].ff;
                                    ans2 = v[i1].
    second + v[i2].second + v[i3].ss + v[i4].ss + v[i5
    ].ss + v[i6].ss + v[i7].ss + v[i8].ss;
                                    bool eh = aprov(
    ans1, ans2);
                                    if(eh)
                                    {
                                        cout << cnt <<
     '\n';
                                        cout << i1 +1
    << " " << i2 +1 << " " << i3+1 << " " << i4+1 << "
     " << i5+1  << " " << i6 +1 << " " << i7 +1 << " "
     << i8+1 << "\n";
                                        return 0;
                                    }
                                }
                            }
                        }

                    }
                }
            }

        }

    }

    cout << "Reprovado\n";

  return 0;
}
```

1.18 Two Sat.cpp

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

using namespace std;
#define int long long
#define pii pair<int,int>
#define ll long long
#define vi vector<int>
#define pb push_back
#define endl "\n"
#define input(x) for (auto &it : x) cin >> it;
#define output(x) for (auto &it : x) cout << it << '
    ';
#define sws std::ios::sync_with_stdio(false); cin.tie(
    NULL); cout.tie(NULL);
#define ff first
#define ss second
 const int INF = 0x3f3f3f3f3f;
const long double PI = acos(-1);
const int MAX = 1004;

int n;
int componente[MAX];
vector<int> adj[MAX];
vector<int> adj2[MAX];
vector<int> saida;
int vis[MAX];
bool ans[MAX];

void dfs(int u)
{
    vis[u] = 1;
    for(auto v : adj[u])
    {
        if(!vis[v])
        {
            dfs(v);
        }
    }
    saida.pb(u);
}

void dfs2(int u, int c)
{
    vis[u] = 2;
    componente[u] = c;
    for(auto v : adj2[u])
    {
        if(vis[v] == 1) dfs2(v, c);
    }
}
void add(int a, bool na, int b, bool nb)
{
    a = 2*(abs(a)-1) ^ na;
    b = 2*(abs(b)-1) ^ nb;
    int neg_a = a ^ 1;
    int neg_b = b ^ 1;
    adj[neg_a].pb(b);
    adj2[b].pb(neg_a);

}
bool possible()
{
    for(int i = 0; i < n; i++)
    {
        if(componente[2*i] == componente[2*i+1])
    return false;
        ans[i] = componente[2*i + 1] < componente[2*i
    ];
    }
    return true;
}
void solve()
{
    cin >> n;
    vector<vector<int>> m(3, vector<int>(n));
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < n; j++)
        {
            int x;
            cin >> x;
            m[i][j] = x;
        }
    }

   for(int i = 0; i < n; i++)
   {
        add(m[0][i], m[0][i] > 0, m[1][i], m[1][i] >
    0);
        add(m[0][i], m[0][i] > 0, m[2][i], m[2][i] >
    0);

        add(m[1][i], m[1][i] > 0, m[0][i], m[0][i] >
    0);
        add(m[1][i], m[1][i] > 0, m[2][i], m[2][i] >
    0);

        add(m[2][i], m[2][i] > 0, m[0][i], m[0][i] >
    0);
        add(m[2][i], m[2][i] > 0, m[1][i], m[1][i] >
    0);
   }

    // for(int i = 0; i < 2*n +2; i++)
    // {
    //     cout << i << ": ";
    //     for(auto x : adj[i])
    //     {
    //         cout << x << " ";
    //     }
    //     cout << endl;
    // }
    //  return;

    for(int i = 0; i < 2*n; i++)
    {
        if(!vis[i])
        {
            dfs(i);
        }
    }
```

```
    int c = 0;
    for(int i = saida.size() - 1; i >= 0; i--)
    {
        if(vis[saida[i]] == 1)
        {
            c++;
            dfs2(saida[i], c);
        }
    }

    bool resp = possible();

    cout << (resp? "YES\n" : "NO\n");

    return;
}
int32_t main()
{   sws

    int t;
    //t = 1;
    cin >> t;

    while(t--)
    {
        memset(vis,0,sizeof(vis));
        memset(componente, 0 , sizeof(componente));
        for(int i = 0; i < MAX; i++)adj[i].clear();
        for(int i = 0; i < MAX; i++)adj2[i].clear();
        saida.clear();
        solve();
    }
    return 0;
}
```