# Std Code Library(Temp version1.0)

tingyx

Nanjing University of Science and Technology

August 15, 2024

## Contents

## 分块

1. 分块 I(区间修改, 区间查询)

```cpp
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 2e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

int n, m, sq;
int v[N], bl[N], tag[N], ans[N];

void update(int a, int b){
    for (int i = a; i <= min(bl[a] * sq, b); ++i) {
        ans[bl[a]] -= (v[i] ^ tag[bl[a]]);
        v[i] ^= 1;
        ans[bl[a]] += (v[i] ^ tag[bl[a]]);
    }
    if (bl[a] != bl[b]){
        for (int i = (bl[b] - 1) * sq + 1; i <= b; ++i) {
            ans[bl[b]] -= (v[i] ^ tag[bl[b]]);
            v[i] ^= 1;
            ans[bl[b]] += (v[i] ^ tag[bl[b]]);
        }
    }
    for (int i = bl[a] + 1; i <= bl[b] - 1; ++i){
        tag[i] ^= 1;
        ans[i] = sq - ans[i];
    }
}

ll query(int a, int b){
    ll res = 0;
    for (int i = a; i <= min(bl[a] * sq, b); ++i) {
        res += (v[i] ^ tag[bl[a]]);
    }
    if (bl[a] != bl[b]){
        for (int i = (bl[b] - 1) * sq + 1; i <= b; ++i) {
            res += (v[i] ^ tag[bl[b]]);
        }
    }
    for (int i = bl[a] + 1; i <= bl[b] - 1; ++i) res += ans[i];
    return res;
}

void solve(){
    cin >> n >> m;
    sq = sqrt(n);
    for (int i = 1; i <= n; ++i) bl[i] = (i - 1) / sq + 1;
    for (int i = 1; i <= m; ++i){
        int op, l, r;
        cin >> op >> l >> r;
        if (op == 0) update(l, r);
        else cout << query(l, r) << endl;
    }
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

2. 分块 II

给出一个长为 $n$ 的数列，以及 $n$ 个操作，操作涉及区间加法，询问区间内小于某个值 $x$ 的元素个数。

```cpp
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 2e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

int n, blo;
int v[N], bl[N], atag[N];
vector<int> ve[2005];

void reset(int x)
{
    ve[x].clear();
    for (int i = (x - 1) * blo + 1; i <= min(x * blo, n); i++)
        ve[x].push_back(v[i]);
    sort(ve[x].begin(), ve[x].end());
}

void add(int a, int b, int c)
{
    for (int i = a; i <= min(bl[a] * blo, b); i++)
        v[i] += c;
    reset(bl[a]);
    if (bl[a] != bl[b])
    {
        for (int i = (bl[b] - 1) * blo + 1; i <= b; i++)
            v[i] += c;
        reset(bl[b]);
    }
    for (int i = bl[a] + 1; i <= bl[b] - 1; i++)
        atag[i] += c;
}

int query(int a, int b, int c)
{
    int ans = 0;
    for (int i = a; i <= min(bl[a] * blo, b); i++)
        if (v[i] + atag[bl[a]] < c)
            ans++;
    if (bl[a] != bl[b])
        for (int i = (bl[b] - 1) * blo + 1; i <= b; i++)
            if (v[i] + atag[bl[b]] < c)
                ans++;
    for (int i = bl[a] + 1; i <= bl[b] - 1; i++)
    {
        int x = c - atag[i];
        ans += lower_bound(ve[i].begin(), ve[i].end(), x) - ve[i].begin();
    }
    return ans;
}

int main()
{
    cin >> n;
    blo = sqrt(n);
    for (int i = 1; i <= n; i++)
        cin >> v[i];
    for (int i = 1; i <= n; i++)
    {
        bl[i] = (i - 1) / blo + 1;
        ve[bl[i]].push_back(v[i]);
    }
    for (int i = 1; i <= bl[n]; i++)
        sort(ve[i].begin(), ve[i].end());
    for (int i = 1; i <= n; i++)
    {
        int f, a, b, c;
        cin >> f >> a >> b >> c;
        if (f == 0)
            add(a, b, c);
        if (f == 1)
            printf("%d\n", query(a, b, c * c));
    }
    return 0;
}
```

3. 线性 RMQ

```
/**
    线性 RMQ
 */

#include <bits/stdc++.h>
#include <limits>
#define endl '\n'
using namespace std;
typedef long long ll;

const int N = 5e4 + 10;
const int M = 2e4 + 10;
const int L = 80;
const int mod = 998244353;
const int inf32 = 0x3f3f3f3f;
const ll inf64 = 4e18;

int a[N + M];
int highbit[M];
int stmax[M][L], stmin[M][L];
int premax[M][L], premin[M][L];
int sufmax[M][L], sufmin[M][L];
int quemax[M][L], quemin[M][L];
int stackmax[L], stackmin[L];

void solve(){
    int n, q;
    cin >> n >> q;
    int B = int(log2(n)); // 块的大小
    int S = (n - 1) / B + 1; // 块的个数
    for (int b = 0; b < S; ++b) stmin[b][0] = inf32;
    for (int i = 0; i < n; ++i){
        cin >> a[i];
        stmin[i / B][0] = min(stmin[i / B][0], a[i]);
        stmax[i / B][0] = max(stmax[i / B][0], a[i]);
    }
    for (int b = S - 1; b >= 0; b--){
        for (int k = 1; b + (1 << k) - 1 < S; ++k){
            stmin[b][k] = min(stmin[b][k - 1], stmin[b + (1 << (k - 1))][k - 1]);
            stmax[b][k] = max(stmax[b][k - 1], stmax[b + (1 << (k - 1))][k - 1]);
        }
    }
    for (int b = 0; b < S; ++b){
        int be = b * B;
        premin[b][0] = premax[b][0] = a[be];
        for (int k = 1; k < B; ++k){
            premin[b][k] = min(premin[b][k - 1], a[be + k]);
            premax[b][k] = max(premax[b][k - 1], a[be + k]);
        }
        sufmin[b][B - 1] = sufmax[b][B - 1] = a[be + B - 1];
        for (int k = B - 2; k >= 0; --k){
            sufmin[b][k] = min(sufmin[b][k + 1], a[be + k]);
            sufmax[b][k] = max(sufmax[b][k + 1], a[be + k]);
        }
    }
    for (int b = 0; b < S; ++b){
        int be = b * B;
        int spmin = 0, nowmin = 0;
        int spmax = 0, nowmax = 0;
        for (int i = 0; i < B; ++i){
            while (spmin && a[be + stackmin[spmin]] > a[be + i]) nowmin ^= 1 << stackmin[spmin--];
            while (spmax && a[be + stackmax[spmax]] < a[be + i]) nowmax ^= 1 << stackmax[spmax--];
            quemin[b][i] = (nowmin ^= 1 << (stackmin[++spmin] = i));
            quemax[b][i] = (nowmax ^= 1 << (stackmax[++spmax] = i));
        }
    }
    for (int i = 2; i <= S; ++i) highbit[i] = highbit[i >> 1] + 1;

    while(q --) {
        int l, r;
        cin >> l >> r;
        l--, r--;
        int L = l / B, R = r / B;
        int li = l % B, ri = r % B;

        int mn = inf32, mx = 0;
        if(L == R) {
            mn = min(mn, a[l + __builtin_ctz(quemin[R][ri] >> li)]);
            mx = max(mx, a[l + __builtin_ctz(quemax[R][ri] >> li)]);
        }
        else {
            mn = min(mn, sufmin[L][li]);
            mn = min(mn, premin[R][ri]);
            mx = max(mx, sufmax[L][li]);
            mx = max(mx, premax[R][ri]);
            int len = R - L - 1;
            int k = highbit[len];
            if(len) {
                mn = min(mn, stmin[L + 1][k]);
                mn = min(mn, stmin[R - (1 << k)][k]);
                mx = max(mx, stmax[L + 1][k]);
                mx = max(mx, stmax[R - (1 << k)][k]);
            }
        }
        cout << mx - mn << endl;
    }
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

## 二分图

1.Graph Coloring I(奇环判断染色)

```
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 3e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

vector<int> G[N];

bool col[N], err;
int st[N], pos[N], top;

void dfs(int x, int fa, int c){
    if (!err && pos[x] && pos[x] < top && col[x] != c){
        err = 1;
```

```cpp
            cout << top - pos[x] + 1 << endl;
            for (int i = pos[x]; i <= top; ++i)
                cout << st[i] << " \n"[i == top];
        }
        if (err || pos[x]) return;
        st[++top] = x;
        pos[x] = top;
        col[x] = c;
        for (auto y : G[x]){
            if (y == fa) continue;
            dfs(y, x, c ^ 1);
        }
        --top;
}

void solve(){
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= m; ++i){
        int u, v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    dfs(1, 0, 0);
    if (!err){
        cout << 0 << endl;
        for (int i = 1; i <= n; ++i) cout << col[i] << " \n"[i == n];
    }
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

2. 增广路算法 (矩阵游戏)

矩阵进行行列交换，能否把对角线都变成 1

```cpp
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 2e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

struct augment_path
{
    vector<vector<int>> g;
    vector<int> pa; // 匹配
    vector<int> pb;
    vector<int> vis; // 访问
    int n, m;        // 两个点集中的顶点数量
    int dfn;         // 时间戳记
    int res;         // 匹配数

    augment_path(int _n, int _m) : n(_n), m(_m)
    {
        assert(0 <= n && 0 <= m);
        pa = vector<int>(n, -1);
        pb = vector<int>(m, -1);
        vis = vector<int>(n);
        g.resize(n);
        res = 0;
        dfn = 0;
    }

    void add(int from, int to)
    {
        assert(0 <= from && from < n && 0 <= to && to < m);
        g[from].push_back(to);
    }

    bool dfs(int v)
    {
        vis[v] = dfn;
        for (int u : g[v])
        {
            if (pb[u] == -1)
            {
                pb[u] = v;
                pa[v] = u;
                return true;
            }
        }
        for (int u : g[v])
        {
            if (vis[pb[u]] != dfn && dfs(pb[u]))
            {
                pa[v] = u;
                pb[u] = v;
                return true;
            }
        }
        return false;
    }

    int solve()
    {
        while (true)
        {
            dfn++;
            int cnt = 0;
            for (int i = 0; i < n; i++)
            {
                if (pa[i] == -1 && dfs(i))
                {
                    cnt++;
                }
            }
            if (cnt == 0)
            {
                break;
            }
            res += cnt;
        }
        return res;
    }
};

void solve()
{
    int n;
    cin >> n;
    augment_path ap(n, n);
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j){
            int x;
            cin >> x;
            if (x) ap.g[i].push_back(j);
        }
    int res = ap.solve();
    if (res == n){
        cout << "Yes" << endl;
        return;
    }
    cout << "No" << endl;
}

signed main()
{
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    cin >> t;
    while (t--)
        solve();
    return 0;
}
```

3. 宿舍的假期 (简略版 AG)

```cpp
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 2e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

struct augment_path
{
    vector<vector<int>> g;
    vector<int> pa; // 匹配
    vector<int> pb;
    vector<int> vis; // 访问
    int n, m;         // 两个点集中的顶点数量
    int dfn;          // 时间戳记
    int res;          // 匹配数

    augment_path(int _n, int _m) : n(_n), m(_m)
    {
        assert(0 <= n && 0 <= m);
        pa = vector<int>(n, -1);
        pb = vector<int>(m, -1);
        vis = vector<int>(n);
        g.resize(n);
        res = 0;
        dfn = 0;
    }

    void add(int from, int to)
    {
        assert(0 <= from && from < n && 0 <= to && to < m);
        g[from].push_back(to);
    }

    bool dfs(int v)
    {
        vis[v] = dfn;
        for (int u : g[v])
        {
            if (pb[u] == -1)
            {
                pb[u] = v;
                pa[v] = u;
                return true;
            }
        }
        for (int u : g[v])
        {
            if (vis[pb[u]] != dfn && dfs(pb[u]))
            {
                pa[v] = u;
                pb[u] = v;
                return true;
            }
        }
        return false;
    }

    int solve()
    {
        while (true)
        {
            dfn++;
            int cnt = 0;
            for (int i = 0; i < n; i++)
            {
                if (pa[i] == -1 && dfs(i))
                {
                    cnt++;
                }
            }
            if (cnt == 0)
            {
                break;
            }
            res += cnt;
        }
        return res;
    }
};

void solve()
{
    int n;
    cin >> n;
    augment_path ap(n, n);
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j){
            int x;
            cin >> x;
            if (x) ap.g[i].push_back(j);
        }
    int res = ap.solve();
    if (res == n){
        cout << "Yes" << endl;
        return;
    }
    cout << "No" << endl;
}

signed main()
{
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    cin >> t;
    while (t--)
        solve();
    return 0;
}
```

## Persistent Data Structure

1. 区间第 K 小

```cpp
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 2e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

int root[N];
int a[N], b[N];
struct PersistentTree{
    int ls[N << 5], rs[N << 5], sum[N << 5], tot = 0;
    //建树
    inline void build(int & rt, int l ,int r){
        rt = ++tot;
        if (l == r) {
            sum[rt] = 0;
            return;
        }
        int mid = l + r >> 1;
        build(ls[rt], l, mid);
        build(rs[rt], mid + 1, r);
        sum[rt] = sum[ls[rt]] + sum[rs[rt]];
    }
    //单点修改
    inline void update(int & rt, int old, int l, int r, int p, int k){
        rt = ++tot;
        ls[rt] = ls[old], rs[rt] = rs[old], sum[rt] = sum[old] + k;
        if (l == r) return;
        int mid = l + r >> 1;
        if (p <= mid) update(ls[rt], ls[old], l, mid, p, k);
        else update(rs[rt], rs[old], mid + 1, r, p, k);
    }
    //区间查询
    inline int query(int old, int rt, int l, int r, int kth){
        if (l == r) return l;
        int mid = (l + r) >> 1;
        int res = sum[ls[rt]] - sum[ls[old]];
        if (kth <= res) return query(ls[old], ls[rt], l, mid, kth);
        else return query(rs[old], rs[rt], mid + 1, r, kth - res);
    }
}T;

void solve(){
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
        b[i] = a[i];
    }
    sort(b + 1, b + 1 + n);
    int cnt = 1;
    for (int i = 2; i <= n; ++i){
        if (b[i] != b[cnt])
            b[++cnt] = b[i];
    }
    T.build(root[0], 1, cnt);
    for (int i = 1; i <= n; ++i){
        int p = lower_bound(b + 1, b + cnt + 1, a[i]) - b;
        T.update(root[i], root[i - 1], 1, cnt, p, 1);
    }
    for (int i = 1; i <= m; ++i){
        int l, r, k;
        cin >> l >> r >> k;
        int idx = T.query(root[l - 1], root[r], 1, cnt, k);
        cout << b[idx] << endl;
    }
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

2. 最大异或和

```cpp
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 6e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

int a[N], s[N];
struct Trie
{
    int cnt, rt[N], ch[N * 33][2], val[N * 33];
    void insert(int o, int lst, int v)
    {
        for (int i = 28; i >= 0; i--)
        {
            val[o] = val[lst] + 1; // 在原版本的基础上更新
            if ((v & (1 << i)) == 0)
            {
                if (!ch[o][0])
                    ch[o][0] = ++cnt;
                ch[o][1] = ch[lst][1];
                o = ch[o][0];
                lst = ch[lst][0];
            }
            else
            {
                if (!ch[o][1])
                    ch[o][1] = ++cnt;
                ch[o][0] = ch[lst][0];
                o = ch[o][1];
                lst = ch[lst][1];
            }
        }
        val[o] = val[lst] + 1;
    }

    int query(int o1, int o2, int v)
    {
        int ret = 0;
        for (int i = 28; i >= 0; i--)
        {
            int t = ((v & (1 << i)) ? 1 : 0);
            if (val[ch[o1][!t]] - val[ch[o2][!t]])
                ret += (1 << i), o1 = ch[o1][!t],
                                 o2 = ch[o2][!t]; // 尽量向不同的地方跳
            else
                o1 = ch[o1][t], o2 = ch[o2][t];
        }
        return ret;
    }
} st;

void solve()
{
    int n, q;
    cin >> n >> q;
    for (int i = 1; i <= n; ++i) cin >> a[i], s[i] = s[i - 1] ^ a[i];
    for (int i = 1; i <= n; ++i){
        st.rt[i] = ++st.cnt;
        st.insert(st.rt[i], st.rt[i - 1], s[i]);
    }
    while (q--){
        char op;
        cin >> op;
        if (op == 'A'){
            ++n;
            cin >> a[n];
            s[n] = s[n - 1] ^ a[n];
            st.rt[n] = ++st.cnt;
            st.insert(st.rt[n], st.rt[n - 1], s[n]);
        }else{
            int l, r, x;
            cin >> l >> r >> x;
            l--, r--;
            if (l == 0) cout << max(s[n] ^ x, st.query(st.rt[r], st.rt[0], s[n] ^ x)) << endl;
            else cout << st.query(st.rt[r], st.rt[l - 1], s[n] ^ x) << endl;
        }
    }
}

signed main()
{
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    // cin >> t;
    while (t--)
        solve();
    return 0;
}
```

## Tree Dp

1. 树的平衡点

```
function<void(int, int)> dfs = [&](int u, int fa){
    sz[u] = 1;
    for (auto v : G[u]){
        if (v == fa) continue;
        dfs(v, u);
        sz[u] += sz[v];
        int mx = max(sz[u], n - sz[u]);
        if (mx <= mn){
            mn = mx;
            id = min(u, id);
        }
    }
};
```

2. 树的最小点覆盖 (最少的点覆盖所有边)

```
void dp(int u) {
    bool fg = 0;
    for(int i = h[u]; ~i; i = nex[i]) {
        int j = v[i];
        fg = 1;
        dp(j);
        f[u][0] += f[j][1];
        f[u][1] += min(f[j][0], f[j][1]);
    }
    f[u][1] += 1;
    if(!fg) {
        f[u][0] = 0; f[u][1] = 1;
    }
}
```

3. 树的最小支配集 (最少的点覆盖所有点)

f[i][0] 选 i 且 i 及 i 的子树都被覆盖了 f[i][1] 不选 i 且 i 被其儿子覆盖 f[i][2] 不选 i 且 i 被其父亲覆盖 (儿子可选可不选)

```
void dfs(int u, int fa){
    f[u][0] = 1; f[u][1] = f[u][2] = 0;
    bool ok = false;
    int tmp = inf32;
    for (auto v : G[u]){
        if (v == fa) continue;
        dfs(v, u);
        f[u][2] += min(f[v][1], f[v][0]);
        f[u][0] += min({f[v][0], f[v][1], f[v][2]});
        if (f[v][0] <= f[v][1]){
            ok = true;
            f[u][1] += f[v][0];
        }else{
            f[u][1] += f[v][1];
            tmp = min(tmp, f[v][0] - f[v][1]);
        }
    }
    if (!ok) f[u][1] += tmp;
}
```

4. 树的最大独立集 (选定的任意两点之间无边)

```
function<void(int, int)> dfs = [&](int u, int fa)
{
    dp[u][1] = h[u];
    for (auto v : G[u])
    {
        if (v == fa)
            continue;
        dfs(v, u);
        dp[u][0] += max(dp[v][1], dp[v][0]);
        dp[u][1] += dp[v][0];
    }
};
```

5. 树上背包 (最多不超过 $m$ 条边)

```
#include <bits/stdc++.h>
#include <cstring>
#include <vector>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 1e2 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

vector<pair<int, int>> G[N];
int n, m, dp[N][N], sz[N], tmp[N];

void dfs(int u, int fa){
    sz[u] = 1;
    for (auto [v, w] : G[u]){
        if (v == fa) continue;
        dfs(v, u);
        memset(tmp, 0, sizeof tmp);
        for (int i = sz[u] - 1; i >= 0; --i){
            for (int j = sz[v] - 1; j >= 0; --j){
                int nxt = i + j + 1;
                if (nxt <= m){
                    tmp[nxt] = max(tmp[nxt], dp[u][i] + dp[v][j] + w);
                }
            }
        }
        sz[u] += sz[v];
        for (int i = 0; i <= sz[u] - 1; ++i)
            dp[u][i] = max(dp[u][i], tmp[i]);
    }
}

void solve(){
    cin >> n >> m;
    for (int i = 1; i < n; ++i){
        int u, v, w;
        cin >> u >> v >> w;
        G[u].emplace_back(v, w);
        G[v].emplace_back(u, w);
    }
    dfs(1, 0);
    int ans = 0;
    for (int i = 1; i <= m; ++i){
        ans = max(ans, dp[1][i]);
    }
    cout << ans << endl;
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

6.2022CCPC-A(树上背包) 爱丽丝想在公园里找到她丢失的猫。

爱丽丝想在公园里找到她丢失的猫。

公园是一棵有根的树，由 $n$ 个顶点组成。顶点的编号从 1 到 $n$，根顶点为 1。

爱丽丝现在位于顶点 1。她知道猫已经从顶点 1 跑到了树的某片叶子上，而且没有顶点被访问超过一次。叶子是没有子顶点的顶点。

每个顶点上都有一个监视器。顶点 $i$ 上的监视器可以观察到猫是否访问了顶点 $i$，以及猫去了哪个顶点 (如果顶点 $i$ 不是叶子)。爱丽丝需要花费 $a_i$ 秒来检查 $i_{th}$ 监视器的数据。

爱丽丝也可以自己搜索一些叶子。搜索 $i$ 个叶子需要 $t_i$ 秒。请注意，$i$ 是顶点的计数，而不是顶点的标签。

帮助爱丽丝确定要检查哪些监视器和搜索哪些树叶，以便唯一确定猫的位置，并尽可能减少所需的总时间。请注意，要检查的监视器和要搜索的树叶应该在一开始就决定好，之后不得更改。

找出最短的时间。

```
#include <bits/stdc++.h>
#include <vector>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 2e3 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

vector<int> G[N];
ll sz[N], a[N], t[N];
ll dp[N][N][2], g[N][N];
ll tmp1[N][2], tmp2[N];


void dfs(int u, int fa){
    if (G[u].size() == 1 && fa){
        sz[u] = 1;
        dp[u][0][0] = dp[u][1][1] = 0;
        dp[u][1][0] = a[u];
        return;
    }
    dp[u][0][0] = 0;
    g[u][0] = 0;
    for (auto v : G[u]){
        if (v == fa) continue;
        dfs(v, u);
        memset(tmp1, 0x3f, sizeof tmp1);
        memset(tmp2, 0x3f, sizeof tmp2);
        for (int i = sz[u]; i >= 0; --i){
            for (int j = sz[v]; j >= 0; --j){
                int nxt = i + j;
                tmp1[nxt][0] = min(tmp1[nxt][0], dp[u][i][0] + dp[v][j][0]);
                tmp1[nxt][1] = min({tmp1[nxt][1], dp[u][i][0] + dp[v][j][1], dp[u][i][1] + dp[v][j][0]});
                tmp2[nxt] = min(tmp2[nxt], g[u][i] + min(dp[v][j][0], dp[v][j][1]));
            }
        }
        sz[u] += sz[v];
        memcpy(dp[u], tmp1, sizeof tmp1);
        memcpy(g[u], tmp2, sizeof tmp2);
    }
    for (int i = 1; i <= sz[u]; ++i)
        dp[u][i][0] = min(dp[u][i][0], g[u][i] + a[u]);
}

void solve(){
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    for (int i = 1; i <= n; ++i) cin >> t[i];
    for (int i = 1; i <= n; ++i) sz[i] = 0;
    for (int i = 1; i <= n; ++i) G[i].clear();
    for (int i = 1, u, v; i < n; ++i){
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    memset(dp, 0x3f, sizeof dp);
    memset(g, 0x3f, sizeof g);
    if (n == 1) {
        cout << 0 << endl;
        return;
    }
    ll ans = inf64;
    dfs(1, 0);
    for (int i = 1; i <= sz[1]; ++i){
        ans = min(ans, t[sz[1] - i] + min(dp[1][i][0], dp[1][i][1]));
    }
    cout << ans << endl;
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    cin >> t;
    while(t--) solve();
    return 0;
}
```

```
#include <bits/stdc++.h>
#include <vector>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 2e3 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

vector<int> G[N];
ll sz[N], a[N], t[N];
ll dp[N][N][2], g[N][N];

void dfs(int u, int fa){
    if (G[u].size() == 1 && fa){
        sz[u] = 1;
        dp[u][0][0] = dp[u][1][1] = 0;
        dp[u][1][0] = a[u];
        return;
    }
    dp[u][0][0] = 0;
    g[u][0] = 0;
    for (auto v : G[u]){
        if (v == fa) continue;
        dfs(v, u);
```

```cpp
        for (int i = sz[u]; i >= 0; --i){
            for (int j = sz[v]; j >= 0; --j){
                int nxt = i + j;
                dp[u][nxt][0] = min(dp[u][nxt][0], dp[u][i][0] + dp[v][j][0]);
                dp[u][nxt][1] = min({dp[u][nxt][1], dp[u][i][0] + dp[v][j][1], dp[u][i][1] + dp[v][j][0]});
                g[u][nxt] = min(g[u][nxt], g[u][i] + min(dp[v][j][0], dp[v][j][1]));
            }
        }
        sz[u] += sz[v];
    }
    for (int i = 1; i <= sz[u]; ++i)
        dp[u][i][0] = min(dp[u][i][0], g[u][i] + a[u]);
}

void solve(){
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    for (int i = 1; i <= n; ++i) cin >> t[i];
    for (int i = 1; i <= n; ++i) sz[i] = 0;
    for (int i = 1; i <= n; ++i) G[i].clear();
    for (int i = 1, u, v; i < n; ++i){
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    memset(dp, 0x3f, sizeof dp);
    memset(g, 0x3f, sizeof g);
    if (n == 1) {
        cout << 0 << endl;
        return;
    }
    ll ans = inf64;
    dfs(1, 0);
    for (int i = 1; i <= sz[1]; ++i){
        ans = min(ans, t[sz[1] - i] + min(dp[1][i][0], dp[1][i][1]));
    }
    cout << ans << endl;
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    cin >> t;
    while(t--) solve();
    return 0;
}
```

7. 树联通点集 (换根)

```cpp
#include <bits/stdc++.h>
#include <vector>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 1e6 + 10;
const ll mod = 1e9 + 7;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

int n;
ll f[N], ans[N];
vector<int> G[N];

ll qmi(ll a, ll b = mod - 2){
    ll res = 1;
    while (b){
        if (b & 1) res = res * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return res;
}

void dfs1(int u, int fa){
    f[u] = 1;
    for (auto v : G[u]){
        if (v == fa) continue;
        dfs1(v, u);
        f[u] = f[u] * (f[v] + 1) % mod;
    }
}

void dfs2(int u, int fa){
    if (!fa) ans[u] = f[u];
    else if ((f[u] + 1) % mod == 0){
        dfs1(u, u);
        ans[u] = f[u];
    }else ans[u] = (ans[fa] % mod * qmi(f[u] + 1) + 1) % mod * f[u] % mod;
    for (auto v : G[u]){
        if (v == fa) continue;
        dfs2(v, u);
    }
}

void solve(){
    cin >> n;
    for (int i = 1; i < n; ++i){
        int u, v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    dfs1(1, 1);
    dfs2(1, 0);
    for (int i = 1; i <= n; ++i) cout << ans[i] << endl;
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

8. 树划分联通块大小小于等于 k(树上背包)

另一种背包写法

```
#include <bits/stdc++.h>
#include <cstring>
#include <vector>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 2e3 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

int n, k, sz[N];
vector<int> G[N];
ll dp[N][N], ans = 0; //dp[i][j] i 所在的联通块大小为 j

void dfs(int u, int fa){
    sz[u] = dp[u][1] = 1;
    for (auto v : G[u]){
        if (v == fa) continue;
        dfs(v, u);
```

```cpp
        ll sum = 0;
        for (int i = 1; i <= sz[v] && i <= k; ++i) sum = (sum + dp[v][i]) % mod;
        for (int i = min(sz[u], k); i >= 1; --i){
            for (int j = min(sz[v], k); j >= 1; --j){
                int nxt = i + j;
                if (nxt <= k){
                    dp[u][nxt] = (dp[u][nxt] + dp[u][i] * dp[v][j]) % mod;
                }
            }
            dp[u][i] = (dp[u][i] * sum) % mod;
        }
        sz[u] += sz[v];
    }
}

void solve(){
    cin >> n >> k;
    for (int i = 1; i < n; ++i){
        int u, v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    dfs(1, 0);
    for (int i = 1; i <= k && i <= sz[1]; ++i) ans = (ans + dp[1][i]) % mod;
    cout << ans << endl;
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

8. 树上子链 (点权和最大)

给定一棵树 T， 树 T 上每个点都有一个权值。定义一颗树的子链的大小为：这个子链上所有结点的权值和。请在树 T 中找出一条最大的子链并输出。

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
vector<ll> G[maxn];
ll w[maxn], dis[maxn], ans = -1e18;

void solve(){
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i){
        cin >> w[i];
    }
    for (int i = 1; i <= n - 1; ++i){
        int u, v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    function<void(int, int)> dfs = [&](int u, int fa){
        ll tmp = 0, mx1 = 0, mx2 = 0;
        for (auto v: G[u]){
            if (v == fa) continue;
            dfs(v, u);
            tmp = dis[v];
            if (tmp >= mx1){
                mx2 = mx1;
                mx1 = tmp;
            }else if (tmp >= mx2){
                mx2 = tmp;
            }
        }
        ans = max(ans, mx1 + mx2 + w[u]);
        dis[u] = mx1 + w[u];
    };
    dfs(1, 0);
    cout << ans << endl;
}

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

9.Nim Cheater(树剖优化 dp)

```cpp
#include <bits/stdc++.h>
#define endl '\n'

using ll = long long;

constexpr int N = 4e4 + 10;
constexpr int mod = 998244353;

using namespace std;

int dp[16390];
int f[32][16390];

void solve(){
    int n;
    cin >> n;
    vector<int> a(n + 1), b(n + 1), fa(n + 1), ask(n + 1), ans(n + 1);
    vector<vector<int>> G(n + 1);
    int lst = 0, rt = 0;
    for (int i = 1; i <= n; ++i) {
        string s;
        cin >> s;
        if (s[0] == 'A'){
            cin >> a[i] >> b[i];
            if (rt == 0) rt = i, lst = i;
            else{
                G[lst].push_back(i);
                fa[i] = lst;
                lst = i;
            }
            ask[i] = i;
        }else{
            lst = fa[lst];
            ask[i] = lst;
        }
    }
    vector<int> sz(n + 1), son(n + 1);

    function<void(int)> dfs1 = [&](int u) {
        sz[u] = 1;
        for (auto v : G[u]){
            dfs1(v);
            sz[u] += sz[v];
            if (sz[v] > sz[son[u]]) son[u] = v;
        }
    };

    dfs1(rt);

    function<void(int, int, int)> dfs2 = [&](int u, int num, int sum){
        memcpy(dp, f[num], sizeof(dp));
        for (int i = 0; i < 16384; ++i){
            dp[i] = max(dp[i], f[num][i ^ a[u]] + b[u]);
        }
        ans[u] = sum - dp[0];
        memcpy(f[num], dp, sizeof(f[num]));
        for (auto v : G[u]){
            if (v == son[u]) continue;
            memcpy(f[num + 1], f[num], sizeof(f[num]));
            dfs2(v, num + 1, sum + b[v]);
        }
        if (son[u])
            dfs2(son[u], num, sum + b[son[u]]);
    };

    memset(f, -0x3f, sizeof(f));
    f[0][0] = 0;
    dfs2(rt, 0, b[rt]);
    for (int i = 1; i <= n; ++i) cout << ans[ask[i]] << endl;
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

## 树状数组

1. 二维偏序

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

constexpr int MAXN = 1e7 + 5;
int sum[MAXN], ans[MAXN];
vector<pair<int, int>> vec;
vector<tuple<int, int, int, int>> q;

inline int lowbit(int x) { return x & (-x); }

void add(int pos, int x)
{
    for (; pos < MAXN; pos += lowbit(pos))
        sum[pos] += x;
}

int query_presum(int pos)
{
    int ans = 0;
    for (; pos > 0; pos -= lowbit(pos))
        ans += sum[pos];
    return ans;
}

int main()
{
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n, m, x1, x2, y1, y2;
    cin >> n >> m;
    for (int i = 0; i < n; i++)
    {
        cin >> x1 >> y1, ++x1, ++y1;
        vec.emplace_back(x1, y1);
    }
    sort(vec.begin(), vec.end());
    for (int i = 0; i < m; i++)
    {
        cin >> x1 >> y1 >> x2 >> y2;
        ++x1, ++y1, ++x2, ++y2;
        q.emplace_back(x1 - 1, y1 - 1, 1, i);
        q.emplace_back(x1 - 1, y2, -1, i);
        q.emplace_back(x2, y1 - 1, -1, i);
        q.emplace_back(x2, y2, 1, i);
    }
    sort(q.begin(), q.end());
    int cur = 0;
    for (auto [x, y, c, id] : q)
    {
        while (cur < n && vec[cur].first <= x)
            add(vec[cur].second, 1), ++cur;
        ans[id] += c * query_presum(y);
    }
    for (int i = 0; i < m; i++)
        cout << ans[i] << "\n";

    return 0;
}
```

## 数位 DP

1.XOR SUM

路易斯喜欢整数。他将非负整数序列 $A=[a_1,a_2,\dots,a_k]$ 的值定义为以下公式（$\oplus$ 表示位排他性-或）:

$$\sum_{i=1}^{k}\sum_{j=1}^{i-1} a_i \oplus a_j$$

他想知道有多少个不同的序列 $A$ 满足以下条件:

- $A$ 的长度是 $k$ 。
- $A$ 的值是 $n$ 。
- $0 \le a_i \le m\ (1 \le i \le k)$.

当且仅当存在一个索引 $i$ 时，$[a_1,\dots,a_k]$、$[b_1,\dots,b_k]$ 这两个序列被认为是不同的，即 $a_i \ne b_i$ 。告诉路易斯答案模块 $10^9+7$ 。

输入一行包含三个整数 $n$ , $m$ , $k$ , ( $0 \le n \le 10^{15}$ , $0 \le m \le 10^{12}$ , $1 \le k \le 18$ )。

```
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 21;
const ll mod = 1e9 + 7;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;
int C[N][N];

void init(int n){
    int i, j;
    for (C[0][0] = i = 1; i <= n; ++i)
        for (C[i][0] = j = 1; j <= i; ++j)
            C[i][j] = (C[i - 1][j] + C[i - 1][j - 1]) % mod;
}

ll n, m, k;
unordered_map<ll, ll> f[N << 1][N];
ll dp(int now, int lim, ll sum){
    if (sum > n) return 0;
    if (sum + (k >> 1ll) * (k + 1 >> 1ll) * ((1ll << now + 1) - 1) < n) return 0ll;
    if (now < 0) return sum == n;
    if (f[now][lim].count(sum)) return f[now][lim][sum];
    ll res = 0;
    if (m >> now & 1){
        for (int i = 0; i <= lim; ++i) //之前卡上界的个数
            for (int j = 0; j <= k - lim; ++j)
               (res += C[lim][i] * C[k - lim][j] % mod * dp(now - 1, i ,sum + (i + j) * (k - i - j) * (1ll << now)) %
 ↪  mod) %= mod;
    }else{
        for (int j = 0; j <= k - lim; ++j){
           (res += C[lim][0] * C[k - lim][j] % mod * dp(now - 1, lim, sum + j * (k - j) * (1ll << now)) % mod) %= mod;
        }
    }
    return f[now][lim][sum] = res;
}

void solve(){
    cin >> n >> m >> k;
    init(k);
    ll ans = dp(39, k, 0);
    cout << ans << endl;
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

## math

1. 区间筛法

```
#include <bits/stdc++.h>
#define endl '\n'
#define int ll

using ll = long long;

constexpr int N = 1.2e6 + 10;
constexpr int P = 1e5 + 10;
constexpr int M = 108;
constexpr int mod = 998244353;

using namespace std;

bool vis[N];
int prime[P], pcnt, ans;

void init(){
    for (int i = 2; i < N; ++i) if (!vis[i]){
        prime[++pcnt] = i;
        for (int j = 2; j * i < N; ++j){
            vis[i * j] = 1;
        }
    }
}

inline int S(ll x){
    int res = 0;
    while(x) res += x % 10, x /= 10;
    return res;
}

int val[110];

void solve(){
    ll n;
    ans = 0;
```

```cpp
    cin >> n;
    if (n <= M){
        for (int i = 1; i <= n; ++i)
            if (n % i == S(i)) ans++;
        cout << ans << endl;
        return;
    }
    vector<pair<ll, int>> a[M + 1];
    for (int i = 0; i <= M; ++i) val[i] = n - i;
    for (int i = 1; 1ll * prime[i] * prime[i] <= n; ++i){
        if (prime[i] <= M){
            for (int j = 0; j <= M; ++j){
                int cnt = 0;
                while (val[j] % prime[i] == 0){
                    val[j] /=prime[i];
                    cnt++;
                }
                if (cnt) a[j].push_back({prime[i], cnt});
            }
        }else if (n % prime[i] <= M){
            int j = n % prime[i], cnt = 0;
            while (val[j] % prime[i] == 0){
                val[j] /= prime[i];
                cnt++;
            }
            if (cnt) a[j].push_back({prime[i], cnt});
        }
    }
    for (int i = 0; i <= M; ++i){
        if (val[i] > 1) a[i].push_back({val[i], 1});
    }

    using pt = vector<pair<ll, int>>::iterator;
    auto dfs = [&](auto self, ll x, pt s, pt t, int res) -> void {
        if (s == t) {
            if (S(x) == res) ans += (x > res);
            return;
        }
        auto [y, z] = *s;
        ++s, self(self, x, s, t, res);
        for (int i = 1; i <= z; ++i){
            x *= y;
            self(self, x, s, t, res);
        }
        return;
    };

    for (int i = 0; i <= M; ++i){
        dfs(dfs, 1, a[i].begin(), a[i].end(), i);
    }
    cout << ans << endl;
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    init();
    int t = 1;
    cin >> t;
    while(t--) solve();
    return 0;
}
```

2. 矩阵四则运算

```cpp
#include <bits/stdc++.h>
#define endl '\n'

using ll = long long;

constexpr int N = 2e5 + 10;
constexpr int mod = 1e9 + 7;
```

```cpp

using namespace std;

constexpr int SIZE = 100;

struct Matrix {
    ll M[SIZE + 5][SIZE + 5];

    void clear() {memset(M, 0, sizeof(M));}

    void reset() {
        clear();
        for (int i = 1; i <= SIZE; i++) M[i][i] = 1;
    }

    Matrix friend operator * (const Matrix & A, const Matrix & B) {
        Matrix C;
        C.clear();
        for (int i = 1; i <= SIZE; i++) {
            for (int j = 1; j <= SIZE; j++) {
                for (int k = 1; k <= SIZE; k++) {
                    C.M[i][j] = (C.M[i][j] + A.M[i][k] * B.M[k][j]) % mod;
                }
            }
        }
        return C;
    }

    Matrix friend operator + (const Matrix & A, const Matrix & B) {
        Matrix C;
        C.clear();
        for (int i = 1; i <= SIZE; i++) {
            for (int j = 1; j <= SIZE; j++) {
                C.M[i][j] = (A.M[i][j] + B.M[i][j]) % mod;
            }
        }
        return C;
    }

    Matrix friend operator - (const Matrix & A, const Matrix & B) {
        Matrix C;
        C.clear();
        for (int i = 1; i <= SIZE; i++){
            for (int j = 1; j <= SIZE; j++){
                C.M[i][j] = (A.M[i][j] - B.M[i][j] + mod) % mod;
            }
        }
        return C;
    }

    Matrix friend operator ^ (const Matrix & A, ll p) {
        Matrix C;
        C.reset();
        auto tmp = A;
        while (p) {
            if (p & 1) C = C * tmp;
            tmp = tmp * tmp;
            p >>= 1;
        }
        return C;
    }
};

Matrix t3, t4, t5;

Matrix MatrixSum(const Matrix & A, ll c) {
    if (c == 1){
        t3 = A;
        return A;
    }
    Matrix t2 = MatrixSum(A, c / 2ll);
    if (c % 2) {
        t4 = t3 * t3 * A;
        t5 = t3;
        t3 = t4;
        return (t2 + (t2 * t5)) + t3;
    }else{
        t4 = t3 * t3;
        t5 = t3;
        t3 = t4;
        return t2 + (t2 * t5);
    }
}

Matrix f1, f2;

void solve(){
    ll n, m, L, R;
    cin >> n >> m >> L >> R;
    f1.clear(), f2.clear();
    vector<vector<ll>> e(2 * n + 1, vector<ll>(2 * n + 1, 0ll));
    for (int i = 1; i <= n * 2; ++i){
        for (int j = 1; j <= n * 2; ++j) {
            e[i][j] = 0;
        }
    }
    for (int i = 1; i <= m; ++i){
        ll u, v, w;
        cin >> u >> v >> w;
        e[u][v] = (e[u][v] + w) % mod;
    }
    f1.M[1][1] = 1;
    for (int i = 2; i <= n; ++i){
        for (int j = 1; j < i; ++j){
            f1.M[1][i] = (f1.M[1][i] + f1.M[1][j] * e[j][i]) % mod;
        }
    }

    for (int i = 1; i <= n; ++i){
        for (int j = 1; j < n + i; ++j){
            if (j <= n) {
                f2.M[j][i] = (f2.M[j][i] + e[j][n + i]) % mod;
            }else {
                for (int k = 1; k <= n; ++k){
                    f2.M[k][i] = (f2.M[k][i] + f2.M[k][j - n] * e[j - n][i]) % mod;
                }
            }
        }
    }
    ll st = L / n; if (L % n == 0) st--;
    ll ed = R / n; if (R % n == 0) ed--;
    Matrix f3, f4;
    f3.clear(), f4.clear();
    if (!st) {
        f3 = f1;
    }else {
        auto p = f2 ^ st;
        for (int i = 1; i <= n; ++i){
            for (int j = 1; j <= n; ++j){
                f3.M[1][i] = (f3.M[1][i] + f1.M[1][j] * p.M[j][i])   % mod;
            }
        }
    }
    ll ans = 0;
    ll t1 = L % n;
    if (!t1) t1 = n;
    if (st == ed){
        ll t2 = R % n;
        if (!t2) t2 = n;
        for (int i = t1; i <= t2; ++i){
            ans = (ans + f3.M[1][i]) % mod;
        }
```

```cpp
    }else{
        for (int i = t1; i <= n; ++i) ans = (ans + f3.M[1][i]) % mod;
        f4 = f3;
        f3.clear();
        auto p = f2 ^ ed;
        for (int i = 1; i <= n; ++i){
            for (int j = 1; j <= n; ++j){
                f3.M[1][i] = (f3.M[1][i] + f1.M[1][j] * p.M[j][i]) % mod;
            }
        }
        ll t2 = R % n;
        if (!t2) t2 = n;
        for (int i = 1; i <= t2; ++i){
            ans = (ans + f3.M[1][i]) % mod;
        }
        if (st != ed - 1){
            auto p = MatrixSum(f2, ed - st - 1);
            f3.clear();
            for (int i = 1; i <= n; ++i) {
                for (int j = 1; j <= n; ++j){
                    f3.M[1][i] = (f3.M[1][i] + f4.M[1][j] * p.M[j][i]) % mod;
                }
            }
            for (int i = 1; i <= n; ++i){
                ans = (ans + f3.M[1][i]) % mod;
            }
        }
    }
    cout << ans << endl;
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t = 1;
    cin >> t;
    while(t--) solve();
    return 0;
}
```

3. 构造异或方程组非 0 解

```cpp
#include <bits/stdc++.h>
#define endl '\n'

using ll = long long;

constexpr int N = 2e5 + 10;
constexpr int mod = 998244353;

using namespace std;

array<bitset<60>, 60> matrix;
array<ll, 60> ans = {0};

void solve(){
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= m; ++i){
        int u, v;
        cin >> u >> v; u--, v--;
        matrix[u].set(v);
        matrix[v].set(u);
    }

    for (int i = 0, j = 0; i < n; ++i){
        for (int k = j; k < n; ++k){
            if (matrix[k][i]){
                swap(matrix[j], matrix[k]);
                for (int l = 0; l < n; ++l){
                    if (l != j && matrix[l][i]){
                        matrix[l] ^= matrix[j];
                    }
                }
                ++j;
                break;
            }
        }
    }

    for (int i = 0; i < n; ++i){
        if (matrix[i].count() == 1){
            cout << "No" << endl;
            return;
        }
        if (!matrix[i].count()) break;
        int a = 0;
        while (!matrix[i][a]) ++a;
        for (int j = a + 1; j < n; ++j){
            if (matrix[i][j]){
                ans[a] |= 1ll << n - j - 1;
                ans[j] |= 1ll << n - j - 1;
            }
        }
    }

    cout << "Yes" << endl;
    for (int i = 0; i < n; ++i){
        cout << ans[i] + !ans[i] << " \n"[i == n - 1];
    }
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

## 图论

1.(有向图) 强联通分量缩点

```cpp
#include <bits/stdc++.h>
#define endl '\n'

using ll = long long;

constexpr int N = 2e5 + 10;
constexpr int mod = 998244353;

using namespace std;

struct SCC{
    int n, now, cnt;
    vector<vector<int>> ver;
    vector<int> dfn, low, col, id, S;

    SCC(int n) : n(n), ver(n + 1), low(n + 1), id(n + 1){
        dfn.resize(n + 1, - 1);
        col.resize(n + 1, - 1);
        now = cnt = 0;
    }

    void add(int x, int y){
        ver[x].push_back(y);
    }

    void tarjan(int x){
        dfn[x] = low[x] = now++;
        S.push_back(x);
        for (auto y : ver[x]) {
            if (dfn[y] == -1) {
                tarjan(y);
                low[x] = min(low[x], low[y]);
            }else if (col[y] == -1){
                low[x] = min(low[x], dfn[y]);
            }
        }
        if (dfn[x] == low[x]) {
            int pre;
            cnt++;
            id[cnt] = x;
            do {
                pre = S.back();
                col[pre] = cnt;
                S.pop_back();
            }while (pre != x);
        }
    }

    auto work () {
        for (int i = 1; i <= n; ++i){
            if (dfn[i] == -1) tarjan(i);
        }

        vector<int> siz(cnt + 1);
        vector<vector<int>> adj(cnt + 1);

        for (int i = 1; i <= n; ++i){
            siz[col[i]]++;
            for (auto j : ver[i]){
                int x = col[i], y = col[j];
                if (x != y){
                    adj[x].push_back(y);
                }
            }
        }

        //return tuple{cnt, adj, col, sz};
        return tuple{cnt, col, siz, id};
    }
};

void solve(){
    int n, m;
    cin >> n >> m;
    SCC scc(n);
    vector<pair<int, int>> e(n + 1);
    for (int i = 1; i <= m; ++i){
        int u, v;
        cin >> u >> v;
        e[i] = {u, v};
        scc.add(u, v);
    }

    auto [cnt, col, sz, id] = scc.work();

    vector<int> in(n + 1);
    for (int i = 1; i <= m; ++i){
        auto [u, v] = e[i];
        if (col[u] != col[v]){
            in[col[v]]++;
        }
    }
    vector<int> ans;
    // for (int i = 1; i <= cnt; ++i){
    //     cout << id[i] << " \n"[i == cnt];
    // }
    for (int i = 1; i <= cnt; ++i){
        if (in[i] == 0) ans.push_back(id[i]);
    }
```

```cpp
    sort(ans.begin(), ans.end());
    cout << ans.size() << endl;
    for (auto x : ans) {
        cout << x << " ";
    }
    cout << endl;
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

## Segment Tree

1. 线段树 I

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define int ll
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 4e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

ll a[N], sum[N << 2], tag[N << 2];

void push_up(int x) {sum[x] = sum[x << 1] + sum[x << 1 | 1];}

void build(int u, int l, int r){
    tag[u] = 0;
    if (l == r) {sum[u] = a[l]; return;}
    int mid = (l + r) >> 1;
    build(u << 1, l, mid);
    build(u << 1 | 1, mid + 1, r);
    push_up(u);
}

inline void f(int u, int l, int r, int k){
    tag[u] += k;
    sum[u] += k * (r - l + 1);
}

inline void push_down(int u, int l, int r){
    int mid = (l + r) >> 1;
    f(u << 1, l, mid, tag[u]);
    f(u << 1 | 1, mid + 1, r, tag[u]);
    tag[u] = 0;
}

inline void update(int u, int nl, int nr, int l, int r, int k){
    if (nl <= l && r <= nr) {f(u, l, r, k); return;}
    push_down(u, l, r);
    int mid = (l + r) >> 1;
    if (nl <= mid) update(u << 1, nl, nr, l, mid, k);
    if (nr > mid) update(u << 1 | 1, nl, nr, mid + 1, r, k);
    push_up(u);
}

inline int query(int u, int qx, int qy, int l, int r){
    ll res = 0;
    if (qx <= l && r <= qy) return sum[u];
    int mid = l + r >> 1;
    push_down(u, l, r);
    if (qx <= mid) res += query(u << 1, qx, qy, l, mid);
    if (qy > mid) res += query(u << 1 | 1, qx, qy, mid + 1, r);
    return res;
}

void solve(){
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    build(1, 1, n);
    while(m--){
        int op, x, y, k;
        cin >> op >> x >> y;
        if (op == 1) {
            cin >> k;
            update(1, x, y, 1, n, k);
        } else {
            cout << query(1, x, y, 1, n) << endl;
        }
    }
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

2. 线段树 II

```cpp
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 4e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

ll a[N], sum[N << 2], taga[N << 2], tagm[N << 2], p;

void push_up(int x) {sum[x] = (sum[x << 1] + sum[x << 1 | 1]) % p;}

void build(int u, int l, int r){
    taga[u] = 0;
    tagm[u] = 1;
    if (l == r) {sum[u] = a[l]; return;}
    int mid = (l + r) >> 1;
    build(u << 1, l, mid);
    build(u << 1 | 1, mid + 1, r);
    push_up(u);
}

inline void push_down(int u, int l, int r){
    int m = l + r >> 1;
    ll &tm = tagm[u], &ta = taga[u];
    if (tm != 1){
        taga[u << 1] = (taga[u << 1] * tm) % p;
        taga[u << 1 | 1] = (taga[u << 1 | 1] * tm) % p;
        tagm[u << 1] = (tagm[u << 1] * tm) % p;
        tagm[u << 1 | 1] = (tagm[u << 1 | 1] * tm) % p;
        sum[u << 1] = (sum[u << 1] * tm) % p;
        sum[u << 1 | 1] = (sum[u << 1 | 1] * tm) % p;
        tm = 1;
    }
    if (ta){
        taga[u << 1] = (taga[u << 1] + ta) % p;
        taga[u << 1 | 1] = (taga[u << 1 | 1] + ta) % p;
        sum[u << 1] = (sum[u << 1] + ta * (m - l + 1)) % p;
        sum[u << 1 | 1] = (sum[u << 1 | 1] + ta * (r - m)) % p;
        ta = 0;
    }
}

inline void update1(int u, int nl, int nr, int l, int r, int k){
    if (nl <= l && r <= nr) {
        tagm[u] = (tagm[u] * k) % p;
        taga[u] = (taga[u] * k) % p;
        sum[u] = (sum[u] * k) % p;
        return;
    }
    int mid = (l + r) >> 1;
    push_down(u, l, r);
    if (nl <= mid) update1(u << 1, nl, nr, l, mid, k);
    if (nr > mid) update1(u << 1 | 1, nl, nr, mid + 1, r, k);
    push_up(u);
}

inline void update2(int u, int nl, int nr, int l, int r, int k){
    if (nl <= l && r <= nr) {
        taga[u] = (taga[u] + k) % p;
        sum[u] = (sum[u] + k * (r - l + 1)) % p;
        return;
    }
    int mid = (l + r) >> 1;
    push_down(u, l, r);
    if (nl <= mid) update2(u << 1, nl, nr, l, mid, k);
    if (nr > mid) update2(u << 1 | 1, nl, nr, mid + 1, r, k);
    push_up(u);
}

inline int query(int u, int qx, int qy, int l, int r){
    ll res = 0;
    if (qx <= l && r <= qy) return sum[u];
    int mid = l + r >> 1;
    push_down(u, l, r);
    if (qx <= mid) res = (res + query(u << 1, qx, qy, l, mid)) % p;
    if (qy > mid) res = (res + query(u << 1 | 1, qx, qy, mid + 1, r)) % p;
    return res;
}

void solve(){
    int n, m;
    cin >> n >> m >> p;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    build(1, 1, n);
    while(m--){
        int op, x, y, k;
        cin >> op >> x >> y;
        if (op == 1) {
            cin >> k;
            update1(1, x, y, 1, n, k);
        } else if (op == 2){
            cin >> k;
            update2(1, x, y, 1, n, k);
        } else{
            cout << query(1, x, y, 1, n) << endl;
        }
    }
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

3. 扫描线

```
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 2e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

struct line{
    int x1, x2, h, tag;
}li[N];
bool cmp (line a, line b) {return a.h < b.h;}

struct Tree{
    int l, r, len, cnt;
}tr[N * 8];

int d[N], n;

void build(int p, int l, int r){
    tr[p] = {l, r};
    if (l == r) return;
    int mid = l + r >> 1;
    build(p << 1, l, mid);
    build(p << 1 | 1, mid + 1, r);
}

void push_up(int p){
    int l = tr[p].l, r = tr[p].r;
    if (tr[p].cnt) tr[p].len = d[r + 1] - d[l];
    else tr[p].len = tr[p << 1].len + tr[p << 1 | 1].len;
}

void update(int p, int l, int r, int k){
    if (l <= tr[p].l && tr[p].r <= r) {
        tr[p].cnt += k;
        push_up(p);
        return;
    }
    int mid = tr[p].l + tr[p].r >> 1;
    if (l <= mid) update(p << 1, l, r, k);
    if (r > mid) update(p << 1 | 1, l, r, k);
    push_up(p);
}

void solve(){
    cin >> n;
    for (int i = 1; i <= n; ++i){
        int a, b, c, e;
        cin >> a >> b >> c >> e;
        li[i] = {a, c, b, 1};
        li[i + n] = {a, c, e, -1};
        d[i] = a, d[i + n] = c;
    }
    n <<= 1;
    sort(li + 1, li + n + 1, cmp);
    sort(d + 1, d + n + 1);
    int len = unique(d + 1, d + n + 1) - d - 1;
    build(1, 1, len - 1); //只需要 lens - 1 个区间位置即可
    ll ans = 0;
    for (int i = 1; i < n; ++i){
        int x1 = lower_bound(d + 1, d + len + 1, li[i].x1) - d;
```

```cpp
        int x2 = lower_bound(d + 1, d + len + 1, li[i].x2) - d;
        update(1, x1, x2 - 1, li[i].tag);
        ans += 1ll * (li[i + 1].h - li[i].h) * tr[1].len;
    }
    cout << ans << endl;
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

4. 窗口的星星 (另一个版本的扫描线)

对于每组数据:

第一行 3 个整数 $n, W, H$ 表示有 $n$ 颗星星，窗口宽为 $W$，高为 $H$。

接下来 $n$ 行，每行三个整数 $x_i, y_i, l_i$ 表示星星的坐标在 $(x_i, y_i)$，亮度为 $l_i$。

输出 $T$ 个整数，表示每组数据中窗口星星亮度总和的最大值。

```cpp
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 2e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

ll n, w, h, d[N];
struct line{
    ll x1, x2, h, val;
    bool operator < (const line& a) const{
        return (h != a.h) ? h < a.h : val > a.val;
    }
}L[N];

struct SegmentTree{
    ll l, r, mx, add;
}T[N << 2];

void push_up(int p){
    T[p].mx = max(T[p << 1].mx, T[p << 1 | 1].mx);
}

void build(int p, int l, int r){
    T[p].l = l, T[p].r = r;
    T[p].mx = T[p].add = 0;
    if (l == r) return;
    int mid = l + r >> 1;
    build(p << 1, l, mid);
    build(p << 1 | 1, mid + 1, r);
    push_up(p);
}

void push_down(int p){
    if (T[p].add){
        T[p << 1].add += T[p].add;
        T[p << 1 | 1].add += T[p].add;
        T[p << 1].mx += T[p].add;
        T[p << 1 | 1].mx += T[p].add;
        T[p].add = 0;
    }
}

```

```cpp
void update(int p, int ql, int qr, ll val){
    int l = T[p].l, r = T[p].r;
    if (ql <= l && r <= qr){
        T[p].add += val;
        T[p].mx += val;
        return;
    }
    push_down(p);
    int mid = l + r >> 1;
    if (ql <= mid) update(p << 1, ql, qr, val);
    if (qr > mid) update(p << 1 | 1, ql, qr, val);
    push_up(p);
}

void solve(){
    cin >> n >> w >> h;
    for (int i = 1; i <= n; ++i){
        ll x, y, l;
        cin >> x >> y >> l;
        d[(i << 1) - 1] = y;
        d[(i << 1)] = y + h - 1;
        L[(i << 1) - 1] = (line){y, y + h - 1, x, l};
        L[(i << 1)] = (line){y, y + h - 1, x + w - 1, -l};
    }
    n <<= 1;
    sort(d, d + n + 1);
    sort(L + 1, L + n + 1);
    ll cnt = unique(d, d + n + 1) - d - 1;
    for (int i = 1; i <= n; ++i){
        ll x1 = lower_bound(d + 1, d + cnt + 1, L[i].x1) - d;
        ll x2 = lower_bound(d + 1, d + cnt + 1, L[i].x2) - d;
        L[i].x1 = x1, L[i].x2 = x2;
    }
    build(1, 1, cnt);
    ll ans = 0;
    for (int i = 1; i <= n; ++i){
        update(1, L[i].x1, L[i].x2, L[i].val);
        ans = max(ans, T[1].mx);
    }
    cout << ans << endl;
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    cin >> t;
    while(t--) solve();
    return 0;
}
```

5. 二分 + 线段树

输入数据的第一行为两个整数 $n$ 和 $m$， $n$ 表示序列的长度， $m$ 表示局部排序的次数。

第二行为 $n$ 个整数，表示 1 到 $n$ 的一个排列。

接下来输入 $m$ 行，每一行有三个整数 op, $l, r$， op 为 0 代表升序排序， op 为 1 代表降序排序, $l, r$ 表示排序的区间。

最后输入一个整数 $q$，表示排序完之后询问的位置

输出数据仅有一行，一个整数，表示按照顺序将全部的部分排序结束后第 $q$ 位置上的数字。

```cpp
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;

const ll N = 1e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;
```

```cpp

struct Sort
{ // 记录这 m 次排序操作
    int op, l, r;
} q[N];
struct Node
{                   // 线段树
    int l, r;       // sum 记录 01 序列中 1 的个数
    int sum, lazy; // lazy 为懒标记: 1 代表将此段全部变为 1, -1 代表将此段全部变为 0
} tr[N * 4];
int n, m, k, a[N];
void pushup(int u)
{
    tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
}
void pushdown(int u)
{
    if (tr[u].lazy)
    {
        tr[u << 1].lazy = tr[u << 1 | 1].lazy = tr[u].lazy;
        if (tr[u].lazy == 1)
        {
            tr[u << 1].sum = tr[u << 1].r - tr[u << 1].l + 1;
            tr[u << 1 | 1].sum = tr[u << 1 | 1].r - tr[u << 1 | 1].l + 1;
        }
        else
            tr[u << 1].sum = tr[u << 1 | 1].sum = 0;
        tr[u].lazy = 0;
    }
}
void build(int u, int l, int r, int x)
{
    if (l == r)
        tr[u] = {l, r, a[l] >= x, 0}; // 序列中大于等于 x 的数变为 1, 小于 x 的数变为 0
    else
    {
        tr[u] = {l, r};
        int mid = l + r >> 1;
        build(u << 1, l, mid, x), build(u << 1 | 1, mid + 1, r, x);
        pushup(u);
    }
}
int query(int u, int l, int r) // 查询 [l,r] 中 1 的个数
{
    if (l <= tr[u].l && tr[u].r <= r)
        return tr[u].sum;
    pushdown(u);
    int mid = tr[u].l + tr[u].r >> 1;
    int sum = 0;
    if (mid >= l)
        sum = query(u << 1, l, r);
    if (mid < r)
        sum += query(u << 1 | 1, l, r);
    return sum;
}
void update(int u, int l, int r, int c) // 将 [l,r] 区间中的数全部变为 c
{
    if (l <= tr[u].l && tr[u].r <= r)
    {
        tr[u].sum = c * (tr[u].r - tr[u].l + 1);
        tr[u].lazy = c ? 1 : -1;
    }
    else
    {
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (mid >= l)
            update(u << 1, l, r, c);
        if (mid < r)
            update(u << 1 | 1, l, r, c);
        pushup(u);
    }
}
bool queryPoint(int u, int x) // 查询 x 位置上的数是否为 1
{
    if (tr[u].l == tr[u].r)
        return tr[u].sum;
    pushdown(u);
    int mid = tr[u].l + tr[u].r >> 1;
    if (x <= mid)
        return queryPoint(u << 1, x);
    else
        return queryPoint(u << 1 | 1, x);
}
bool check(int mid) // 检查此答案值是否合法
{
    build(1, 1, n, mid); // 用此答案值建树
    for (int i = 1; i <= m; i++)
    {
        int op = q[i].op, l = q[i].l, r = q[i].r; // 对 [l,r] 区间进行排序
        int cnt = query(1, l, r);                 // 查询 [l,r] 中 1 的个数
        if (cnt == 0 || cnt == r - l + 1)
            continue; // 如果区间中的数全部相同，那么不需要进行排序
        if (op)
        {
            update(1, l, cnt + l - 1, 1);
            update(1, cnt + l, r, 0);
        }
        else
        {
            update(1, l, r - cnt, 0);
            update(1, r - cnt + 1, r, 1);
        }
    }
    return queryPoint(1, k); // 所有操作完成后查看 k 位置上的数是否为 1
}

void solve()
{
    cin >> n >> m;
    for (int i = 1; i <= n; ++i)
        cin >> a[i];
    for (int i = 1; i <= m; ++i)
    {
        cin >> q[i].op >> q[i].l >> q[i].r;
    }
    cin >> k;
    int l = 1, r = n;
    while (l < r)
    {
        int mid = l + r + 1 >> 1;
        if (check(mid))
            l = mid;
        else
            r = mid - 1;
    }
    cout << l << endl;
}

signed main()
{
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    // cin >> t;
    while (t--)
        solve();
    return 0;
}
```

6. 线段树合并 (HDU2024 暑期多校 1003)

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

```cpp
#define endl '\n'
using namespace std;
typedef long double db;
typedef long long ll;
typedef unsigned long long ull;

const ll N = 5e5 + 10;
const ll mod = 998244353;
const ll inf32 = 0x3f3f3f3f;
const ll inf64 = 5e18;

vector<int> G[N];
int p[N], rk[N];

int rt[N], ls[N * 60], rs[N * 60], cnt;
int c[N * 60];
ull t[N * 60];
ull cur = 0;
ull ans[N], S1[N], S2[N], a[N];


void push_up(int p){
    t[p] = t[ls[p]] + t[rs[p]];
    c[p] = c[ls[p]] + c[rs[p]];
}

void upd(int l, int r, int &p, int x, ull v) {
    if (!p)
        p = ++cnt;
    if (l == r) {
        c[p] = 1;
        t[p] = v;
        return;
    }
    int m = (l + r) >> 1;
    if (x <= m)
        upd(l, m, ls[p], x, v);
    else
        upd(m + 1, r, rs[p], x, v);
    push_up(p);
}

int merge(int x, int y, ull prev1, ull prev2) {
    if (!x || !y) {
        cur += prev1 * t[y];
        cur += prev2 * t[x];
        return x + y;
    }
    int z = ++cnt;
    ls[z] = merge(ls[x], ls[y],prev1, prev2);
    rs[z] = merge(rs[x], rs[y], prev1 + c[ls[x]], prev2 + c[ls[y]]);
    push_up(z);
    return z;
}

void dfs(int u, int fa){
    upd(0, N, rt[u], rk[u], a[u] * a[u]);
    S1[u] = a[u];
    S2[u] = a[u] * a[u];
    for (auto v : G[u]){
        if (v == fa) continue;
        dfs(v, u);
        S1[u] += S1[v];
        S2[u] += S2[v];
        ans[u] += ans[v];
        cur = 0;
        rt[u] = merge(rt[u], rt[v], 0, 0);
        ans[u] += cur;
    }
}

```

```cpp
void solve(){
    int n;
    cin >> n;
    for (int i = 1; i < n; ++i){
        int u, v;
        cin >> u >> v;
        --u, --v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    for (int i = 0; i < n; ++i) cin >> a[i];
    for (int i = 0; i < n; ++i) p[i] = i;
    sort(p, p + n, [&](int i, int j){return a[i] < a[j];});
    for (int i = 0; i < n; ++i) rk[p[i]] = i;
    dfs(0, -1);
    for (int i = 0; i < n; ++i){
        ans[i] *= 2;
        ans[i] += S2[i];
        ans[i] -= S1[i] * S1[i];
    }
    ull res = 0;
    for (int i = 0; i < n; ++i) res ^= ans[i];
    cout << res << endl;
}

signed main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

7. 动态开点线段树

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

const int N = 100010;
#define int long long

struct node {
    int l, r;
    int add, sum;
} tr[N << 1];
// 正常线段树，这里不开 4 倍大小会 RE

int n, m, idx, root;
int a[N];

void pushup(int p) {
    tr[p].sum = tr[tr[p].l].sum + tr[tr[p].r].sum;
}

void pushdown(int p, int l, int r) {
    if(tr[p].add) {
        int mid = l + r >> 1;
        tr[tr[p].l].sum += (mid - l + 1) * tr[p].add, tr[tr[p].l].add += tr[p].add;
        tr[tr[p].r].sum += (r - mid) * tr[p].add, tr[tr[p].r].add += tr[p].add;
        tr[p].add = 0;
    }
}

void build(int &p, int l, int r) {
    if(!p) p = ++idx;
    if(l == r) { tr[p].sum = a[l]; return ;}
    int mid = l + r >> 1;
    build(tr[p].l, l, mid), build(tr[p].r, mid + 1, r);
    pushup(p);
}

void modify(int &p, int l, int r, int ql, int qr, int k) {
    if(!p) p = ++idx;
    if(l >= ql && r <= qr) {
        tr[p].sum += (r - l + 1) * k;
        tr[p].add += k;
        return ;
    }
    pushdown(p, l, r);
    int mid = l + r >> 1;
    if(ql <= mid) modify(tr[p].l, l, mid, ql, qr, k);
    if(qr > mid) modify(tr[p].r, mid + 1, r, ql, qr, k);
    pushup(p);
}

int query(int p, int l, int r, int ql, int qr) {
    if(l >= ql && r <= qr) { return tr[p].sum; }
    int mid = l + r >> 1;
    pushdown(p, l, r);
    int v = 0;
    if(ql <= mid) v = query(tr[p].l, l, mid, ql, qr);
    if(qr > mid) v += query(tr[p].r, mid + 1, r, ql, qr);
    return v;
}

signed main() {
    cin >> n >> m;
    for (int i = 1; i <= n; i++) cin >> a[i];

    build(root, 1, n);

    int op, x, y, k;
    while(m--) {
        cin >> op >> x >> y;
        if(op == 1) {
            cin >> k;
            modify(root, 1, n, x, y, k);
        } else {
            cout << query(root, 1, n, x, y) << endl;
        }
    }
    return 0;
}
```

8. 线段树 (单点修改、区间最值)

```cpp
#include <bits/stdc++.h>
#include <vector>
#define endl '\n'
using namespace std;
typedef long long ll;

const int N = 2e5 + 10;
const int mod = 998244353;
const int inf32 = 0x3f3f3f3f;
const ll inf64 = 4e18;

struct Node
{
    int l, r;
    ll v1, v2;
}tr[N * 4];

ll a[N], b[N];

void pushup(int u)
{
    tr[u].v1 = max(tr[u << 1].v1, tr[u << 1 | 1].v1);
    tr[u].v2 = min(tr[u << 1].v2, tr[u << 1 | 1].v2);
}

void build(int u, int l, int r)
{
    tr[u] = {l, r};
    if (l == r) {
        tr[u].v1 = tr[u].v2 = b[l];
        return;
    }
    int mid = l + r >> 1;
    build(u << 1, l, mid), build(u << 1 | 1, mid + 1, r);
    pushup(u);
}

ll query1(int u, int l, int r)
{
    if (tr[u].l >= l && tr[u].r <= r) return tr[u].v1;
    int mid = tr[u].l + tr[u].r >> 1;
    ll v = -inf64;
    if (l <= mid) v = query1(u << 1, l, r);
    if (r > mid) v = max(v, query1(u << 1 | 1, l, r));
    return v;
}

ll query2(int u, int l, int r)
{
    if (tr[u].l >= l && tr[u].r <= r) return tr[u].v2;
    int mid = tr[u].l + tr[u].r >> 1;
    ll v = inf64;
    if (l <= mid) v = query2(u << 1, l, r);
    if (r > mid) v = min(v, query2(u << 1 | 1, l, r));
    return v;
}

void modify(int u, int x, ll v)
{
    if (tr[u].l == x && tr[u].r == x) tr[u].v1 += v, tr[u].v2 += v;
    else
    {
        int mid = tr[u].l + tr[u].r >> 1;
        if (x <= mid) modify(u << 1, x, v);
        else modify(u << 1 | 1, x, v);
        pushup(u);
    }
}


void solve(){
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i)
        cin >> a[i];
    for (int i = 2; i <= n; ++i) {
        b[i] = a[i] - a[i - 1];
    }
    build(1, 1, n);
    int q;
    cin >> q;
    while (q--){
        int op, l, r;
        ll x;
        cin >> op >> l >> r;
        if (op == 1) {
            cin >> x;
            modify(1, l, x);
            modify(1, r + 1, -x);
        }else if (op == 2){
            if (l == r){
                cout << 1 << endl;
            }else{
                ll mx = query1(1, l + 1, r);
                ll mn = query2(1, l + 1, r);
                if (mx == 0 && mn == 0){
                    cout << 1 << endl;
                }else{
                    cout << 0 << endl;
                }
            }
        }else if (op == 3){
            if (l == r){
                cout << 1 << endl;
            }else{
                ll mn = query2(1, l + 1, r);
                cout << (mn > 0) << endl;
            }
        }else if (op == 4){
            if (l == r){
                cout << 1 << endl;
            }else{
                ll mx = query1(1, l + 1, r);
                cout << (mx < 0) << endl;
            }
        }else {
            int ql = l + 1, qr = r + 1, ok = 0;
            while (ql < qr){
                int mid = (ql + qr) >> 1;
                ll mn = query2(1, l + 1, mid);
                if (mn > 0) {
                    ok = mid;
                    ql = mid + 1;
                }else{
                    qr = mid;
                }
            }
            if (!ok) {
                cout << 0 << endl;
            }else{
                if (ok >= r) {
                    cout << 0 << endl;
                }else{
                    ll mx = query1(1, ok + 1, r);
                    if (mx < 0) {
                        cout << 1 << endl;
                    }else{
                        cout << 0 << endl;
                    }
                }
            }
        }
    }
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

9. 单点修改，区间最大字段和

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

#define int long long

const int maxn = 1001000;
int n, m;
int ans;

struct tree{
    int lmax; // 当前区间最大前缀和
    int rmax; // 当前区间最大后缀和
    int maxx; // 当前区间最大子段和
    int sum; // 当前区间的和
}t[4 * maxn];
```

```cpp

void push_up(int rt){
    t[rt].sum = t[rt << 1].sum + t[rt << 1 | 1].sum;
    // 当前区间的和: 左子树的和 + 右子树的和
    t[rt].rmax = max(t[rt << 1 | 1].rmax, t[rt << 1 | 1].sum + t[rt << 1].rmax);
    // 当前区间的最大后缀和: 右子树的最大后缀和 or 右子树的和 + 左子树的最大后缀和
    t[rt].lmax = max(t[rt << 1].lmax, t[rt << 1].sum + t[rt << 1 | 1].lmax);
    // 当前区间的最大前缀和: 左子树的最大前缀和 or 左子树的和 + 右子树的最大前缀和
    t[rt].maxx = max(t[rt << 1].rmax + t[rt << 1 | 1].lmax, max(t[rt << 1].maxx, t[rt << 1 | 1].maxx));
    // 当前区间的最大子段和: 左子树的最大子段和 or 右子树的最大子段和 or 左子树的最大后缀和 + 右子树的最大前缀和
}

void build(int rt, int l, int r){
    if(l == r){
        cin >> t[rt].maxx;
        t[rt].lmax = t[rt].rmax = t[rt].sum = t[rt].maxx;
        return;
    }
    int mid = l + r >> 1;
    build(rt << 1, l, mid);
    build(rt << 1 | 1, mid + 1, r);
    push_up(rt);
}

void update(int rt, int l, int r, int x, int y){
    if(l == r){
        t[rt].lmax = t[rt].rmax = t[rt].maxx = t[rt].sum = y;
        return ;
    }
    int mid = l + r >> 1;
    if(mid >= x) update(rt << 1, l, mid, x, y);
    else update(rt << 1 | 1, mid + 1, r, x, y);
    push_up(rt);
}

tree query(int rt, int l, int r, int x, int y){
    if(l >= x && r <= y) return t[rt]; // 区间完全覆盖, 直接返回该节点
    int mid = l + r >> 1;
    if(y <= mid) return query(rt << 1, l, mid, x, y); // 只在左区间, 直接查询左区间
    else if(x > mid) return query(rt << 1 | 1, mid + 1, r, x, y); // 只在右区间, 直接查询右区间
    else{
        tree res_l = query(rt << 1, l, mid, x, y);
        tree res_r = query(rt << 1 | 1, mid + 1, r, x, y);
        tree res;
        // res_l 记录左覆盖区间, res_r 记录右覆盖区间, 合并后得到 res
        // 用 push_up 同样的方式更新 res
        res.sum = res_l.sum + res_r.sum;
        res.lmax = max(res_l.sum + res_r.lmax, res_l.lmax);
        res.rmax = max(res_r.rmax, res_r.sum + res_l.rmax);
        res.maxx = max(max(res_l.maxx, res_r.maxx), res_l.rmax + res_r.lmax);
        return res;
    }
}

signed main(){
    cin >> n >> m;
    build(1, 1, n);
    int opt, x, y;
    while(m--){
        cin >> opt >> x >> y;
        if(opt == 1){
            if(x > y) swap(x, y);
            tree ans = query(1, 1, n, x, y);
            cout << ans.maxx << '\n';
        }
        else update(1, 1, n, x, y);
    }
    return 0;
}
```

10. 半群 (一次函数嵌套)

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define int ll
using namespace std;
typedef long long ll;

const int N = 5e5 + 10;
const int mod = 998244353;
const int inf32 = 0x3f3f3f3f;
const ll inf64 = 4e18;

int n, m, K[N],b[N];

struct node{
    int l, r;
    ll mul, sum;
}t[N << 2];

node unite(node x, node y){
    node ans;
    ans.l = x.l, ans.r = y.r;
    ans.mul = x.mul * y.mul % mod;
    ans.sum = (x.sum * y.mul % mod + y.sum) % mod;
    return ans;
}

void push_up(int u){
    t[u] = unite(t[u << 1], t[u << 1 | 1]);
}

void build(int u, int l, int r){
    if (l == r){
        t[u].l = t[u].r = l;
        t[u].sum = b[l];
        t[u].mul = K[l];
        return;
    }
    int mid = l + r >> 1;
    build(u << 1, l, mid);
    build(u << 1 | 1, mid + 1, r);
    push_up(u);
}

void modify(int u, int x, int K, int B){
    if (t[u].l == t[u].r)
    {
        t[u].sum = B;
        t[u].mul = K;
        return;
    }
    int mid = t[u].l + t[u].r >> 1;
    if (x <= mid) modify(u << 1, x, K, B);
    else modify(u << 1 | 1, x, K, B);
    push_up(u);
}

node query(int u, int l, int r){
    if (t[u].l == l && t[u].r == r) return t[u];
    int mid = t[u].l + t[u].r >> 1;
    if (r <= mid) return query(u << 1, l, r);
    else if (l > mid) return query(u << 1 | 1, l, r);
    else return unite(query(u << 1, l, mid), query(u << 1 | 1, mid + 1, r));
}

void solve(){
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> K[i] >> b[i];
    build(1, 1, n);
    for (int i = 1; i <= m; ++i){
        int op;
        cin >> op;
        if (op == 0){
            int p, c, d;
            cin >> p >> c >> d;
            p++;
            modify(1, p, c, d);
        }else{
            int l, r, x;
            cin >> l >> r >> x;
            l++;
            auto ans = query(1, l, r);
            cout << (ans.mul * x % mod + ans.sum) % mod << endl;
        }
    }
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

11. 树上半群修改，路径查询

```cpp
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long long ll;

const int N = 2e5 + 10;
const int mod = 998244353;
const int inf32 = 0x3f3f3f3f;
const ll inf64 = 4e18;

struct Info {
    int a1, b1, a2, b2;
};

Info operator + (const Info &l , const Info &r) {
    Info ret;
    //信息合并
    ret.a1 = (1LL * l.a1 * r.a1 % mod); //向下的
    ret.b1 = (1LL * r.a1 * l.b1 % mod + r.b1) % mod;
    ret.a2 = (1LL * l.a2 * r.a2 % mod); // 向上的
    ret.b2 = (1LL * l.a2 * r.b2 % mod + l.b2) % mod;
    return ret;
}

struct node {
    int a, b;
};

node operator + (const node &l , const node &r) {
    node ret;
    //信息合并
    ret.a = (1LL * l.a * r.a % mod);
    ret.b = (1LL * r.a * l.b % mod + r.b) % mod;
    return ret;
}

Info t[N << 2];
int a[N], b[N], sz[N], top[N], dep[N], in[N], dfn[N], f[N];

struct Segment{
    int n;

    void push_up(int u) {
        t[u] = t[u << 1] + t[u << 1 | 1];
    }

    void build(int u, int l, int r){
        if (l == r){
            t[u].a1 = t[u].a2 = a[dfn[l]];
            t[u].b1 = t[u].b2 = b[dfn[l]];
            return;
        }
        int mid = l + r >> 1;
        build(u << 1, l ,mid);
        build(u << 1 | 1, mid + 1, r);
        push_up(u);
    }

    void modify(int u, int l, int r, int p, const Info & v){
        if (l == r){
            t[u] = v;
            return;
        }
        int mid = l + r >> 1;
        if (p <= mid) modify(u << 1, l, mid, p, v);
        else  modify(u << 1 | 1, mid + 1, r, p, v);
        push_up(u);
    }

    void modify(int u, int l, int r, int ql, int qr, const Info & v){
        if (l == ql && r == qr){
            t[u] = v;
            return;
        }
        int mid = l + r >> 1;
        if (qr <= mid) modify(u << 1, l, mid, ql, qr, v);
        else if (ql > mid) modify(u << 1 | 1, mid + 1, r, ql, qr, v);
        else {
            modify(u << 1, l, mid, ql, mid, v);
            modify(u << 1 | 1, mid + 1, r, mid + 1, qr, v);
        }
        push_up(u);
    }

    Info query(int rt, int l, int r, int ql, int qr) {
        if (l == ql && r == qr) {
            return t[rt];
        }
        int mid = l + r >> 1;
        if (qr <= mid) return query(rt << 1 , l , mid , ql , qr);
        else if (ql > mid) return query(rt << 1 | 1 , mid + 1 , r , ql , qr);
        else {
            return query(rt << 1 , l , mid , ql , mid) + query(rt << 1 | 1 , mid + 1 , r , mid + 1 , qr);
        }
    }

    Info query(int ql, int qr) {
        if (ql > qr){
            return Info {1, 0, 1, 0};
        }
        return query(1, 1, n, ql, qr);
    }
}tree;

void solve(){
    int n, q;
    cin >> n >> q;
    for (int i = 1; i <= n; ++i){
        cin >> a[i] >> b[i];
    }
    vector<vector<int>> G(n + 1);
    for (int i = 1; i < n; ++i){
        int u, v;
        cin >> u >> v;
        u++, v++;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    auto dfs1 = [&](auto self, int u, int fa) -> void {
        if (fa) G[u].erase(find(G[u].begin(), G[u].end(), fa));
        sz[u]++;
        for (auto &v : G[u]){
            dep[v] = dep[u] + 1;
            f[v] = u;
            self(self, v, u);
            sz[u] += sz[v];
            if (sz[v] > sz[G[u][0]]){
                swap(G[u][0], v);
            }
        }
    };

    int tot = 0 ;
    auto dfs2 = [&](auto self, int u, int fa) -> void {
        in[u] = ++tot;
        dfn[in[u]] = u;
        for (auto & v : G[u]){
            top[v] = (v == G[u][0] ? top[u] : v);
            self(self, v, u);
        }
    };

    dep[1] = 0;
    dfs1(dfs1, 1, 0);
    top[1] = 0;
    dfs2(dfs2, 1, 0);
    tree.n = n;
    tree.build(1, 1, n);

    // 向上的
    auto path1 = [&](auto self, int x, int y) -> node{
        node ans = {1, 0};
        while(top[x] != top[y]){
            if (dep[top[x]] < dep[top[y]]) std::swap(x, y);
            Info res = tree.query(in[top[x]], in[x]);
            ans = (ans + node{res.a2, res.b2});
            x = f[top[x]];
        }
        if (dep[x] > dep[y]) std::swap(x, y);
        Info res = tree.query(in[x] + 1, in[y]);
        ans = (ans + node{res.a2, res.b2});
        return ans;
    };

    auto path2 = [&](auto self, int x, int y) -> node{
        node ans = {1, 0};
        while(top[x] != top[y]){
            if (dep[top[x]] < dep[top[y]]) std::swap(x, y);
            Info res = tree.query(in[top[x]], in[x]);
            ans = (node{res.a1, res.b1} + ans);
            x = f[top[x]];
        }
        if (dep[x] > dep[y]) std::swap(x, y);
        Info res = tree.query(in[x], in[y]);
        ans = (node{res.a1, res.b1} + ans);
        return ans;
    };

    auto lca = [&](auto self, int u, int v) -> int{
        while (top[u] != top[v]) {
            if (dep[top[u]] > dep[top[v]]) {
                u = f[top[u]];
            } else {
                v = f[top[v]];
            }
        }
        return dep[u] < dep[v] ? u : v;
    };

    while(q--){
        int op, l, r, x;
        cin >> op >> l >> r >> x;
        l++;
        if (op == 0){
            tree.modify(1, 1, n, in[l], Info{r, x, r, x});
        }else{
            r++;
            int lc = lca(lca, l, r);
            node ansL = path1(path1, l, lc);
            node ansR = path2(path2, lc, r);
            int ans = (1ll * ansL.a * x + ansL.b) % mod;
            ans = (1ll * ansR.a * ans + ansR.b) % mod;
            cout << ans << endl;
        }
    }
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```

12. 动态开点，权值线段树

```cpp
#include <bits/stdc++.h>
#define endl '\n'

using ll = long long;

constexpr int N = 1e5 + 10;
constexpr int M = 1e7 + 10;
constexpr int mod = 998244353;

using namespace std;

struct SegmentTree{
    int ls, rs;
    int num;
}tr[M];

int n, q, cnt, np1[N], np2[N], f[N];

ll m, k, a[N];

void update(int u, ll l, ll r, ll p, int sum) {
    tr[u].num = sum;
    if (l == r) return;
    ll mid = l + r >> 1;
    if (p <= mid){
        if (!tr[u].ls){
            tr[u].ls = ++cnt;
        }
        update(tr[u].ls, l, mid, p, sum);
    }else{
        if (!tr[u].rs){
            tr[u].rs = ++cnt;
        }
        update(tr[u].rs, mid + 1, r, p, sum);
    }
}

int query(int u, ll l, ll r, ll ql, ll qr) {
    if (ql <= l && r <= qr) return tr[u].num;
    ll mid = l + r >> 1;
    int ans = n + 2;
    if (ql <= mid) {
        ans = min(ans, query(tr[u].ls, l, mid, ql, qr));
    }
    if (qr > mid) {
        ans = min(ans, query(tr[u].rs, mid + 1, r, ql, qr));
    }
    return ans;
}

void solve(){
    cin >> n >> m >> k;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    cnt = 1, tr[0].num = tr[1].num = n + 1;
    for (int i = n; i >= 1; --i){
        np1[i] = query(1, 1ll, m, a[i], min(m, a[i] + k));
        np2[i] = query(1, 1ll, m, max(1ll, a[i] - k), a[i]);
        update(1, 1ll, m, a[i], i);
    }
    cin >> q;
    while (q--){
        int l, r;
        cin >> l >> r;
        for (int i = l; i <= r; ++i) f[i] = 0;
        int ans = 0;
        for (int i = r; i >= l; --i){
            f[i] = 1;
            if (np1[i] <= r) f[i] = max(f[i], f[np1[i]] + 1);
            if (np2[i] <= r) f[i] = max(f[i], f[np2[i]] + 1);
            ans = max(ans, f[i]);
        }
        cout << (r - l + 1 - ans) << endl;
    }
    for (int i = 0; i <= cnt; ++i) tr[i].num = tr[i].ls = tr[i].rs = 0;
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t = 1;
    cin >> t;
    while(t--) solve();
    return 0;
}
```

## 串串

1. 本质不同子序列数量

```
vector<int> dp(n + 1);
map<char, int> vis;
for (int i = pl + 1; i < pr; ++i){
        if(!vis.count(s[i])){
            vis[s[i]] = 1;
            dp[i] = (dp[i - 1] * 2 + 1) % P;
        }else{
            dp[i] = (((dp[i - 1] * 2) % P - dp[vis[s[i]] - 1] + P) % P + P) % P;
        }
        vis[s[i]] = i;
    }
```

2. 子序列自动机

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<i64, i64>
#define tll tuple<i64, i64, i64>
#define all(a) a.begin() + 1, a.end()
using namespace std;
using i64 = long long;
using db = long double;
const i64 N = 1e5 + 10;
const i64 mod = 998244353;
```

```cpp
const i64 inf32 = 1e9;
const i64 inf64 = 5e18;

int typ, n, q, m;
int root[N];
struct PersistentTree{
    int ls[N << 5], rs[N << 5], nxt[N << 5], tot = 0;
    //单点修改
    inline void update(int & rt, int old, int l, int r, int p, int k){
        rt = ++tot;
        ls[rt] = ls[old], rs[rt] = rs[old], nxt[rt] = nxt[old];
        if (l == r){
            nxt[rt] = k;
            return;
        }
        int mid = l + r >> 1;
        if (p <= mid) update(ls[rt], ls[old], l, mid, p, k);
        else update(rs[rt], rs[old], mid + 1, r, p, k);
    }
    //单点查询
    inline int query(int rt, int l, int r, int p){
        if (l == r) return nxt[rt];
        int mid = l + r >> 1;
        if (p <= mid) return query(ls[rt], l, mid, p);
        else return query(rs[rt], mid + 1, r, p);
    }
    //建树
    inline void build(vector<int> a){
        int n = a.size() - 1;
        for (int i = n; i >= 1; --i){
            update(root[i], root[i + 1], 1, m, a[i], i + 1);
        }
    }
}T;

void solve(){
    cin >> typ >> n >> q >> m;
    vector<int> a(n + 1);
    for (int i = 1; i <= n; ++i) cin >> a[i];
    T.build(a);
    while (q--){
        int k;
        cin >> k;
        bool ok = true;
        int rt = 1;
        while (k--){
            int p;
            cin >> p;
            int to = T.query(root[rt], 1, m, p);
            if (!to) ok = false;
            if (to) rt = to;
        }
        if (ok) cout << "Yes" << endl;
        else cout << "No" << endl;
    }
}

int main(){
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while(t--) solve();
    return 0;
}
```