__int128.cpp

```cpp
// __int128 read / print
#include <cstdio>
inline __int128 read() {
    __int128 x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x << 1) + (x << 3) + (ch ^ 48); // x * 10 char->int
        // x = x * 10 + ch - '0';
        ch = getchar();
    }
    return x * f;
}
inline void print(__int128 x) {
    if (x < 0) {
        putchar('-');
        x = -x;
    }
    if (x > 9)
        print(x / 10);
    putchar(x % 10 + '0');
}
```

Kruskal重构树.cpp

```cpp
// Kruskal 重构树 P1967
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 3e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, q, m;
struct Eg {
    int f, t, w;
    Eg(int f, int t, int w) : f(f), t(t), w(w) {};
};
vector<Eg> eg;
vector<int> con[N];
bool cmp(Eg q, Eg w) {
    return q.w > w.w;
}
int fa[N], ans[N], dep[N];
int find(int x) {
    return x == fa[x] ? x : fa[x] = find(fa[x]);
}
void kru() {
    for (int i = 1; i <= n; i++) {
        fa[i] = i;
    }
    sort(eg.begin(), eg.end(), cmp);
    for (auto [f, t, w] : eg) {
        int fx = find(f), tx = find(t);
        if (fx != tx) {
            fa[fx] = fa[tx] = ++n;
            fa[n] = n;
            con[n].push_back(fx);
            con[n].push_back(tx);
            ans[n] = w;
        }
    }
}
int dp[N][32];
void dfs(int a, int f) {
    dp[a][0] = f;
    dep[a] = dep[f] + 1;
    for (int t : con[a]) {
```

```
        if (t == f)
            continue;
        dfs(t, a);
    }
}
void prework() {
    for (int j = 1; j < 30; j++) {
        for (int i = 1; i <= n; i++) {
            dp[i][j] = dp[dp[i][j - 1]][j - 1];
        }
    }
}
int lca(int a, int b) {
    if (find(a) != find(b))
        return 0;
    if (dep[a] < dep[b])
        swap(a, b);
    int k = dep[a] - dep[b];
    for (int j = 0; j < 30; j++) {
        if (k >> j & 1ll) {
            a = dp[a][j];
        }
    }
    for (int j = 29; j >= 0; j--) {
        if (dp[a][j] != dp[b][j]) {
            a = dp[a][j], b = dp[b][j];
        }
    }
    return dp[a][0];
}
bool vis[N];
signed main() {
    ios::sync_with_stdio(false), cin.tie(0);
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        int f, t, w;
        cin >> f >> t >> w;
        eg.push_back({f, t, w});
    }
    kru();
    for (int i = 1; i <= n; i++) {
        if (vis[find(i)])
            continue;
```

```
        vis[find(i)] = true;
        dfs(find(i), find(i));
    }
    prework();
    ans[0] = -1;
    cin >> q;
    while (q--) {
        int a, b;
        cin >> a >> b;
        cout << ans[lca(a, b)] << '\n';
    }
}
```

ST表.cpp

```cpp
// ST表
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, q, a[N], lg[N], f[N][32];
void prework() {
    for (int i = 2; i <= n; i++) {
        lg[i] = lg[i / 2] + 1;
    }
    for (int i = 1; i <= n; i++) {
        f[i][0] = a[i];
    }
    for (int i = 1; i < 31; i++) {
        for (int j = 1; j + (1 << i) - 1 <= n; j++) {
            f[j][i] = max(f[j][i - 1], f[j + (1 << i - 1)][i - 1]);
        }
    }
}
int query(int l, int r) {
    int len = lg[r - l + 1];
    return max(f[l][len], f[r - (1 << len) + 1][len]);
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> q;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    prework();
    while (q--) {
        int l, r;
        cin >> l >> r;
        cout << query(l, r) << '\n';
    }
}
```

tarjan求割点.cpp

```cpp
// tarjan 求割点
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int low[N], cnt, dfn[N];
vector<int> con[N], cut;
void tarjan(int a, bool root) {
    low[a] = dfn[a] = ++cnt;
    int tot = 0;
    for (int t : con[a]) {
        if (!dfn[t]) {
            tarjan(t, false);
            low[a] = min(low[a], low[t]);
            tot += low[t] >= dfn[a];
        } else {
            low[a] = min(low[a], dfn[t]);
        }
    }
    if (tot > root) {
        cut.push_back(a);
    }
}
signed main() {
    int n;
    cin >> n;
    while (scanf("%lld", &n), n) {
        for (int i = 1; i <= n; i++) {
            con[i].clear();
            low[i] = dfn[i] = 0;
        }
        cnt = 0;
        cut.clear();
        int a;
        while (scanf("%lld", &a), a) {
            while (getchar() != '\n') {
                int b;
                scanf("%lld", &b);
                con[a].push_back(b);
                con[b].push_back(a);
```

```
            }
        }
        tarjan(1, true);
        cout << cut.size() << '\n';
    }
}
```

tarjan缩点建DAG图上dp.cpp

```cpp
// tarjan缩点建DAG图上dp
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int dfn[N], low[N], cnt, scc[N], cscc, w[N], cow[N], dp[N], in[N];
bool instk[N];
vector<int> con[N], co[N];
stack<int> stk;
void tarjan(int a) {
    low[a] = dfn[a] = ++cnt;
    stk.push(a);
    instk[a] = true;
    for (int t : con[a]) {
        if (!dfn[t]) {
            tarjan(t);
            low[a] = min(low[a], low[t]);
        } else if (instk[t]) {
            low[a] = min(low[a], dfn[t]);
        }
    }
    if (low[a] == dfn[a]) {
        cscc++;
        int top;
        do {
            top = stk.top();
            stk.pop();
            instk[top] = false;
            scc[top] = cscc;
            cow[cscc] += w[top];  // 累加点权
        } while (top != a);
    }
}
void topo() {
    queue<int> q;
    for (int i = 1; i <= cscc; i++) {
        if (in[i] == 0) {
            dp[i] = cow[i];
            q.push(i);
        }
```

```
    }
    while (q.size()) {
        int a = q.front();
        q.pop();
        for (int t : co[a]) {
            if (--in[t] == 0) {
                q.push(t);
                dp[t] = max(dp[t], dp[a] + cow[t]); // DAG 上 dp
            }
        }
    }
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; i++) {
        cin >> w[i];
    }
    for (int j = 1; j <= m; j++) {
        int f, t;
        cin >> f >> t;
        con[f].push_back(t);
    }
    for (int i = 1; i <= n; i++) {
        if (!dfn[i])
            tarjan(i);
    }
    // 缩点
    for (int i = 1; i <= n; i++) {
        for (int t : con[i]) {
            if (scc[i] != scc[t]) {
                co[scc[i]].push_back(scc[t]);
                in[scc[t]]++;
            }
        }
    }
    topo();
    int ans = 0;
    for (int i = 1; i <= cscc; i++) {
        ans = max(ans, dp[i]);
    }
```

```
    cout << ans;
}
```

Trie树.cpp

```cpp
// Trie
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 3e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, q, con[N][26], cnt;
bool vis[N];
void add(string a) {
    int cur = 1;
    for (int c : a) {
        c -= 'a';
        if (!con[cur][c])
            con[cur][c] = ++cnt;
        cur = con[cur][c];
    }
}
bool ask(string a) {
    int cur = 1;
    for (int c : a) {
        c -= 'a';
        if (!con[cur][c])
            return false;
        cur = con[cur][c];
    }
    return true;
}
signed main() {
    ios::sync_with_stdio(false), cin.tie(0);
    cin >> n;
    while (n--) {
        string a;
        cin >> a;
        add(a);
    }
    cin >> q;
    set<string> s;
    while (q--) {
        string a;
        cin >> a;
        if (ask(a)) {
```

```
            cout << "OK\n";
        } else {
            cout << "NO PREFIX\n";
        }
    }
}
```

带权并查集.cpp

```cpp
// 带权并查集 P1196
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 3e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, fa[N], front[N], num[N];
// front[x] : x 节点所处的位置
// num[x] : 队头 x 所在队列的长度
int find(int x) {
    if (fa[x] == x)
        return x;
    int fx = find(fa[x]);
    front[x] += front[fa[x]];
    return fa[x] = fx;
}
void merge(int x, int y) {
    int fx = find(x), fy = find(y);
    front[fx] += num[fy];
    num[fy] += num[fx];
    num[fx] = 0;
    fa[fx] = fy;
}
int ask(int x, int y) {
    if (find(x) != find(y))
        return -1;
    return abs(front[x] - front[y]) - 1;
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
    for (int i = 1; i <= n; i++) {
        fa[i] = i;
        num[i] = 1;
    }
    while (n--) {
        char op;
        int x, y;
        cin >> op >> x >> y;
        if (op == 'M') {
```

```
            merge(x, y);
        } else {
            cout << ask(x, y) << '\n';
        }
    }
}
```

点分治.cpp

```cpp
// 点分治
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 5e4 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, k, ctr, sz[N], del[N], cnt[N], tmp[N], pt, ans;
vector<int> con[N];
void df(int a, int f) { // 求子树的重心
    sz[a] = 1;
    int mxs = 0;
    for (int t : con[a]) {
        if (t == f or del[t])
            continue;
        df(t, a);
        if (ctr)
            return;
        sz[a] += sz[t];
        mxs = max(mxs, sz[t]);
    }
    mxs = max(mxs, n - sz[a]); // 勿忘
    if (mxs <= n / 2) {
        ctr = a;
        sz[f] = n - sz[a];
    }
}
void dfs(int a, int f, int dep) { // 求答案
    if (dep > k)
        return;
    ans += cnt[k - dep];
    tmp[pt++] = dep;
    for (int t : con[a]) {
        if (t == f or del[t])
            continue;
        dfs(t, a, dep + 1);
    }
}
void run(int a) {
    // 对包含 a 的路径求答案
    // 注意在此清空求答案的临时数组
    // 每次 run 相当于单独求一棵树的答案
```

```
    del[a] = true;
    cnt[0] = 1;
    for (int t : con[a]) {
        if (del[t])
            continue;
        dfs(t, a, 1);
        for (int i = 0; i < pt; i++) {
            cnt[tmp[i]]++;
        }
        pt = 0;
    }
    for (int i = 1; i <= k; i++) {
        cnt[i] = 0;
    }
    // 对不包含 a 的路径求答案
    // 删去 a 点，对子树递归处理
    for (int t : con[a]) {
        if (del[t])
            continue;
        ctr = 0;
        n = sz[t]; // n 会改变，检查在其它地方的使用
        df(t, a);
        run(ctr); // 参数是 ctr
    }
}
signed main() {
    ios::sync_with_stdio(false), cin.tie(0);
    cin >> n >> k;
    for (int i = 1; i < n; i++) {
        int f, t;
        cin >> f >> t;
        con[t].push_back(f);
        con[f].push_back(t);
    }
    df(1, 1);
    run(ctr);
    cout << ans;
}
```

堆优化Dijkstra.cpp

```cpp
// 堆优化Dijkstra
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, m, s, dis[N];
bool vis[N];
struct Eg {
    int t, w;
    Eg(int t, int w) : t(t), w(w){};
    bool operator<(const Eg& oth) const { return w > oth.w; }
};
vector<Eg> con[N];
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> m >> s;
    for (int i = 1; i <= m; i++) {
        int f, t, w;
        cin >> f >> t >> w;
        con[f].push_back(Eg(t, w));
    }
    for (int i = 1; i <= n; i++) {
        dis[i] = INF;
    }
    dis[s] = 0;
    priority_queue<Eg> q;
    q.push(Eg(s, 0));
    while (q.size()) {
        int a = q.top().t;
        q.pop();
        if (vis[a])
            continue;
        vis[a] = true;
        for (auto eg : con[a]) {
            auto [t, w] = eg;
            dis[t] = min(dis[t], dis[a] + w);
            if (!vis[t])
                q.push(Eg(t, dis[t]));
        }
```

```
    }
    for (int i = 1; i <= n; i++) {
        cout << dis[i] << ' ';
    }
}
```

分块+二分求区间高于某值数量.cpp

```cpp
// 分块+二分求区间高于某值数量
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 1e6 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, q, a[N], len[N], st[N], ed[N], bel[N], mark[N], d[N];
void prework() {
    int sq = sqrt(n);
    int dwlen = n / sq;
    for (int i = 1; i <= sq; i++) {
        st[i] = dwlen * (i - 1) + 1;
        ed[i] = i == sq ? n : dwlen * i;
        len[i] = ed[i] - st[i] + 1;
        for (int j = st[i]; j <= ed[i]; j++) {
            a[j] = d[j];
            bel[j] = i;
        }
    }
    for (int i = 1; i <= sq; i++) {
        sort(a + st[i], a + ed[i] + 1);
    }
}
void add(int l, int r, int x) {
    if (bel[l] == bel[r]) {
        for (int i = l; i <= r; i++) {
            d[i] += x;
        }
        for (int i = st[bel[l]]; i <= ed[bel[l]]; i++) {
            a[i] = d[i];
        }
        sort(a + st[bel[l]], a + ed[bel[l]] + 1);
    } else {
        for (int i = l; i <= ed[bel[l]]; i++) {
            d[i] += x;
        }
        for (int i = st[bel[l]]; i <= ed[bel[l]]; i++) {
            a[i] = d[i];
        }
        sort(a + st[bel[l]], a + ed[bel[l]] + 1);
        for (int i = st[bel[r]]; i <= r; i++) {
```

```
            d[i] += x;
        }
        for (int i = st[bel[r]]; i <= ed[bel[r]]; i++) {
            a[i] = d[i];
        }
        sort(a + st[bel[r]], a + ed[bel[r]] + 1);
        for (int i = bel[l] + 1; i <= bel[r] - 1; i++) {
            mark[i] += x;
        }
    }
}
int bs(int l, int r, int x) {  // num of >=x
    while (l <= r) {
        int mid = l + r + 1 >> 1;
        if (a[mid] >= x)
            r = mid - 1;
        else
            l = mid + 1;
    }
    return ed[bel[l]] - r;
}
int query(int l, int r, int x) {
    int ans = 0;
    if (bel[l] == bel[r]) {
        for (int i = l; i <= r; i++) {
            if (d[i] + mark[bel[i]] >= x)
                ans++;
        }
    } else {
        for (int i = l; i <= ed[bel[l]]; i++) {
            if (d[i] + mark[bel[i]] >= x)
                ans++;
        }
        for (int i = st[bel[r]]; i <= r; i++) {
            if (d[i] + mark[bel[i]] >= x)
                ans++;
        }
        for (int i = bel[l] + 1; i <= bel[r] - 1; i++) {
            ans += bs(st[i], ed[i], x - mark[i]);
        }
    }
    return ans;
}
```

```
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> q;
    for (int i = 1; i <= n; i++) {
        cin >> d[i];
    }
    prework();
    while (q--) {
        char op;
        int l, r, x;
        cin >> op >> l >> r >> x;
        if (op == 'M') {
            add(l, r, x);
        } else {
            cout << query(l, r, x) << '\n';
        }
    }
}
```

分块区间改查.cpp

```cpp
// 分块区间改查
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, q, a[N], bel[N], sum[N], mark[N], st[N], ed[N], len[N];
void prework() {
    int sq = sqrt(n);    // number of intervals
    int dwlen = n / sq;  // standard interval lenth
    for (int i = 1; i <= sq; i++) {
        st[i] = dwlen * (i - 1) + 1;
        ed[i] = i == sq ? n : dwlen * i;  // the last interval is longer
        len[i] = ed[i] - st[i] + 1;
        for (int j = st[i]; j <= ed[i]; j++) {
            bel[j] = i;
        }
    }
}
void add(int l, int r, int k) {
    if (bel[l] == bel[r]) {
        for (int i = l; i <= r; i++) {
            a[i] += k;
            sum[bel[i]] += k;
        }
    } else {
        for (int i = l; i <= ed[bel[l]]; i++) {
            a[i] += k;
            sum[bel[i]] += k;
        }
        for (int i = st[bel[r]]; i <= r; i++) {
            a[i] += k;
            sum[bel[i]] += k;
        }
        for (int i = bel[l] + 1; i <= bel[r] - 1; i++) {
            mark[i] += k;
        }
    }
}
int query(int l, int r) {
    int ans = 0;
```

```
    if (bel[l] == bel[r]) {
        for (int i = l; i <= r; i++) {
            ans += a[i] + mark[bel[i]];
        }
    } else {
        for (int i = l; i <= ed[bel[l]]; i++) {
            ans += a[i] + mark[bel[i]];
        }
        for (int i = st[bel[r]]; i <= r; i++) {
            ans += a[i] + mark[bel[i]];
        }
        for (int i = bel[l] + 1; i <= bel[r] - 1; i++) {
            ans += len[i] * mark[i] + sum[i];
        }
    }
    return ans;
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> q;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    prework();
    int sq = sqrt(n);
    for (int i = 1; i <= sq; i++) {
        for (int j = st[i]; j <= ed[i]; j++) {
            sum[i] += a[j];
        }
    }
    while (q--) {
        int op;
        cin >> op;
        if (op == 1) {
            int l, r, k;
            cin >> l >> r >> k;
            add(l, r, k);
        } else {
            int l, r;
            cin >> l >> r;
            cout << query(l, r) << '\n';
        }
```

```
    }
}
```

分数类.cpp

```
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;

struct P {
    int fm{0}, fz{0};
    P() : fm(0), fz(0) {};
    P(int fm, int fz) : fm(fm), fz(fz) {};
    bool operator<(const P& b) const {
        int lcm = fm * b.fm / __gcd(fm, b.fm);
        return fz * lcm / fm < b.fz * lcm / b.fm;
    }
    P operator+(const P& b) {
        int q = fz * b.fm + fm * b.fz;
        int w = fm * b.fm;
        int g = __gcd(q, w);
        q /= g, w /= g;
        return P(w, q);
    }
    P operator*(const P& b) { return P(fm * b.fm, fz * b.fz); }
} dp[N];
```

归并排序求逆序对数.cpp

```cpp
// 归并排序求逆序对数
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int T, ans, t[N], a[N];
int rev(int l, int r) {
    if (l == r)
        return 0;
    int mid = l + r >> 1;
    int ans = rev(l, mid) + rev(mid + 1, r);
    int pl = l, pr = mid + 1, idx = l;
    while (pl <= mid and pr <= r) {
        if (a[pl] < a[pr])
            t[idx++] = a[pl++];
        else
            t[idx++] = a[pr++], ans += mid - pl + 1;
    }
    while (pl <= mid)
        t[idx++] = a[pl++];
    while (pr <= r)
        t[idx++] = a[pr++];
    for (int i = l; i <= r; i++)
        a[i] = t[i];
    return ans;
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> T;
    while (T--) {
        int n;
        cin >> n;
        for (int i = 1; i <= n; i++) {
            cin >> a[i];
        }
        cout << rev(1, n) << '\n';
    }
}
```

快速幂&逆元.cpp

```cpp
// 快速幂 逆元
int ksm(int a, int b, int mod) {
    int res = 1;
    while (b) {
        if (b & 1) {
            res = res * a % mod;
            b--;
        }
        b >>= 1;
        a = a * a % mod;
    }
    return res;
}
int inv(int a, int mod) { // a / b == a * inv(b)
    return ksm(a, mod - 2, mod);
}
```

莫队查询区间不同数的个数.cpp

```cpp
// 莫队查询区间不同数的个数 SP3267
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e6 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int a[N], n, q, bel[N];
struct Q {
    int l, r, idx;
    Q(int l, int r, int idx) : l(l), r(r), idx(idx) {};
    Q() : l(0), r(0), idx(0) {};
    bool operator<(const Q& b) const {
        if (bel[l] != bel[b.l])
            return bel[l] < bel[b.l];
        if (bel[l] % 2 == 1)
            return bel[r] > bel[b.r];
        return bel[r] < bel[b.r];
    }
};
void prework() {
    int sq = sqrt(n);
    for (int i = 0; i < sq; i++) {
        for (int j = 1; j <= sq; j++) {
            bel[i * sq + j] = i + 1;
        }
    }
    for (int i = sq * sq + 1; i <= n; i++) {
        bel[i] = sq;
    }
}
int ans = 0, buc[N], out[N];
void add(int x) {
    if (!buc[a[x]])
        ans++;
    buc[a[x]]++;
}
void del(int x) {
    buc[a[x]]--;
    if (!buc[a[x]])
        ans--;
}
```

```
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    prework();
    vector<Q> qs;
    int cnt = 0;
    cin >> q;
    while (q--) {
        int l, r;
        cin >> l >> r;
        qs.push_back(Q(l, r, ++cnt));
    }
    sort(qs.begin(), qs.end());
    int l = 0, r = 0;
    for (auto [L, R, idx] : qs) {
        while (l < L) {
            del(l++);
        }
        while (l > L) {
            add(--l);
        }
        while (r < R) {
            add(++r);
        }
        while (r > R) {
            del(r--);
        }
        // ans: 区间不同数的个数
        out[idx] = ans;
    }
    for (int i = 1; i <= cnt; i++) {
        cout << out[i] << '\n';
    }
}
```

判重离散化+树状数组求逆序对数.cpp

```cpp
// 判重离散化+树状数组求逆序对数
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 5e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, a[N], tr[N];
struct P {
    int num, idx;
} b[N];
bool cmp(P q, P w) {
    return q.num < w.num;
}
int lowbit(int x) {
    return x & (-x);
}
void add(int pos, int x) {
    while (pos <= n) {
        tr[pos] += x;
        pos += lowbit(pos);
    }
}
int ask(int pos) {
    int ans = 0;
    while (pos) {
        ans += tr[pos];
        pos -= lowbit(pos);
    }
    return ans;
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> b[i].num;
        b[i].idx = i;
    }
    sort(b + 1, b + 1 + n, cmp);
    for (int i = 1; i <= n; i++) {
        if (b[i].num != b[i - 1].num)
```

```
            a[b[i].idx] = i;
        else
            a[b[i].idx] = a[b[i - 1].idx]; // if not unique
    }  // lsh
    int ans = 0;
    for (int i = n; i >= 1; i--) {
        ans += ask(a[i] - 1);
        add(a[i], 1);
    }
    cout << ans;
}
```

强连通分量.cpp

```cpp
// 强连通分量
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
// scc[i]: i 号点所属的强联通分量编号
// cscc: 总强联通分量数
int dfn[N], low[N], cnt, scc[N], cscc;
bool instk[N], scvis[N];
vector<int> con[N], sc[N];
stack<int> stk;
void tarjan(int a) {
    low[a] = dfn[a] = ++cnt;
    stk.push(a);
    instk[a] = true;
    for (int t : con[a]) {
        if (!dfn[t]) {
            tarjan(t);
            low[a] = min(low[a], low[t]);
        } else if (instk[t]) {
            low[a] = min(low[a], dfn[t]);
        }
    }
    if (low[a] == dfn[a]) {
        cscc++;
        int top;
        do {
            top = stk.top();
            stk.pop();
            instk[top] = false;
            scc[top] = cscc;
            sc[cscc].push_back(top);
        } while (top != a);
    }
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n, m;
    cin >> n >> m;
```

```
    for (int j = 1; j <= m; j++) {
        int f, t;
        cin >> f >> t;
        con[f].push_back(t);
    }
    for (int i = 1; i <= n; i++) {
        if (!dfn[i])
            tarjan(i);
    }
    cout << cscc << '\n';
    for (int i = 1; i <= n; i++) {
        if (scvis[scc[i]])
            continue;
        scvis[scc[i]] = true;
        sort(sc[scc[i]].begin(), sc[scc[i]].end());
        for (int t : sc[scc[i]]) {
            cout << t << ' ';
        }
        cout << '\n';
    }
}
```

三分.cpp

```cpp
// 三分 P1883
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 3e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
const double eps = 1e-15;
int n;
double a[N], b[N], c[N];
double f(double x) {
    double mx = -1e9;
    for (int i = 1; i <= n; i++) {
        mx = max(mx, a[i] * x * x + b[i] * x + c[i]);
    }
    return mx;
}
void solve() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> a[i] >> b[i] >> c[i];
    }
    double l = 0, r = 1000, mid;
    while (r - l > eps) {
        double mid = (l + r) / 2;
        if (f(mid - eps) > f(mid + eps)) {
            l = mid;
        } else {
            r = mid;
        }
    }
    cout << fixed << setprecision(4) << f(l) << '\n';
}
signed main() {
    ios::sync_with_stdio(false), cin.tie(0);
    int T;
    cin >> T;
    while (T--) {
        solve();
    }
}
```

树上启发式合并.cpp

```cpp
// 树上启发式合并 洛谷 U41492 统计子树不同颜色数
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
int n, q, c[N], top[N], sz[N], hson[N], ans[N], buc[N], fa[N], diff;
vector<int> con[N];
void df(int a, int f) {
    sz[a] = 1;
    fa[a] = f;
    for (int t : con[a]) {
        if (t == f)
            continue;
        df(t, a);
        sz[a] += sz[t];
        hson[a] = sz[t] > sz[hson[a]] ? t : hson[a];
    }
}
void dt(int a, int f) { // no need
    for (int t : con[a]) {
        if (t == f)
            continue;
        top[t] = t == hson[a] ? top[a] : t;
        dt(t, a);
    }
}
void addSub(int a) {
    diff += buc[c[a]] == 0;
    buc[c[a]]++;
    for (int t : con[a]) {
        if (t == fa[a])
            continue;
        addSub(t);
    }
}
void delSub(int a) {
    diff -= buc[c[a]] == 1;
    buc[c[a]]--;
    for (int t : con[a]) {
        if (t == fa[a])
            continue;
```

```
        delSub(t);
    }
}
void dfs(int a, bool keep) {
    for (int t : con[a]) {
        if (t == fa[a] or t == hson[a])
            continue;
        dfs(t, false);
    }
    if (hson[a])
        dfs(hson[a], true);
    for (int t : con[a]) {
        if (t == fa[a] or t == hson[a])
            continue;
        addSub(t);
    }
    diff += buc[c[a]] == 0;
    buc[c[a]]++;
    ans[a] = diff;
    if (!keep) {
        delSub(a);
    }
}
signed main() {
    ios::sync_with_stdio(false), cin.tie(0);
    cin >> n;
    for (int i = 1; i < n; i++) {
        int f, t;
        cin >> f >> t;
        con[f].push_back(t);
        con[t].push_back(f);
    }
    for (int i = 1; i <= n; i++) {
        cin >> c[i];
    }
    df(1, 1);
    top[1] = 1;
    dt(1, 1);
    dfs(1, 1);
    cin >> q;
    while (q--) {
        int t;
        cin >> t;
```

```
        cout << ans[t] << '\n';
    }
}
```

树状数组.cpp

```cpp
// 树状数组
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, m, tr[N];
int lowbit(int x) {
    return x & (-x);
}
void add(int pos, int x) {
    while (pos <= n) {
        tr[pos] += x;
        pos += lowbit(pos);
    }
}
int ask(int pos) {
    int ans = 0;
    while (pos) {
        ans += tr[pos];
        pos -= lowbit(pos);
    }
    return ans;
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> m;
    for (int i = 1; i <= n; i++) {
        int t;
        cin >> t;
        add(i, t);
    }
    while (m--) {
        int op;
        cin >> op;
        if (op == 1) {
            int x, k;
            cin >> x >> k;
            add(x, k);
        } else {
```

```
            int l, r;
            cin >> l >> r;
            cout << ask(r) - ask(l - 1) << '\n';
        }
    }
}
```

双模字符串哈希.cpp

```cpp
// 双模字符串哈希
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;

struct DoubleHash {
    int P[2] = {131, 13331};
    int mod[2] = {(int)(1e9 + 7), (int)(1e9 + 9)};
    int h[N][2], p[N][2];
    int len;

    void init(string s) {
        s = ' ' + s;
        len = s.size();
        for (int x = 0; x <= 1; x++) {
            p[0][x] = 1;
            for (int i = 1; i < len; i++) {
                p[i][x] = p[i - 1][x] * P[x] % mod[x];
                h[i][x] = (h[i - 1][x] * P[x] % mod[x] + s[i]) % mod[x];
            }
        }
    }
    int get(int l, int r, int x) {
        int tmp = h[r][x] - h[l - 1][x] * p[r - l + 1][x] % mod[x];
        return (tmp % mod[x] + mod[x]) % mod[x];
    }
} hashi;
vector<pair<int, int> > a;

signed main() { // 字符串去重求总数 P3370
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n;
    cin >> n;
    set<int> s;
    for (int i = 1; i <= n; i++) {
        string s;
        cin >> s;
        hashi.init(s);
```

```
        a.push_back({hashi.get(1, s.size(), 0), hashi.get(1, s.size(), 1)});
    }
    sort(a.begin(), a.end());
    for (int i = 1; i < a.size(); i++) {
        auto [n1, n2] = a[i];
        auto [p1, p2] = a[i - 1];
        if (n1 == p1 and n2 == p2) {
            n--;
        }
    }
    cout << n;
}
```

素数筛.cpp

```
bool isnp[N];
vector<int> pri;  // 质数表
void init(int n) {
    for (int i = 2; i <= n; i++) {
        if (!isnp[i])
            pri.push_back(i);
        for (int p : pri) {
            if (p * i > n)
                break;
            isnp[p * i] = 1;
            if (i % p == 0)
                break;
        }
    }
}
```

线段树.cpp

```cpp
// 线段树
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 1e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, a[N], tree[4 * N], mark[4 * N];
void build(int l = 1, int r = n, int pos = 1) {
    if (l == r) {
        tree[pos] = a[l];
    } else {
        int mid = (l + r) / 2;
        build(l, mid, pos * 2), build(mid + 1, r, pos * 2 + 1);
        tree[pos] = tree[pos * 2] + tree[pos * 2 + 1];
    }
}
void push_down(int pos, int cl, int cr) {
    int mid = (cl + cr) / 2;
    tree[pos * 2] += (mid - cl + 1) * mark[pos];
    tree[pos * 2 + 1] += (cr - mid) * mark[pos];
    mark[pos * 2] += mark[pos];
    mark[pos * 2 + 1] += mark[pos];
    mark[pos] = 0;
}
void add(int l, int r, int ad, int pos = 1, int cl = 1, int cr = n) {
    if (cl > r or cr < l) {
        return;
    } else if (cl >= l and cr <= r) {
        tree[pos] += (cr - cl + 1) * ad;
        if (cr > cl)
            mark[pos] += ad;
    } else {
        push_down(pos, cl, cr);
        int mid = (cl + cr) / 2;
        add(l, r, ad, pos * 2, cl, mid);
        add(l, r, ad, pos * 2 + 1, mid + 1, cr);
        tree[pos] = tree[pos * 2] + tree[pos * 2 + 1];
    }
}
int query(int l, int r, int pos = 1, int cl = 1, int cr = n) {
    if (cl > r or cr < l) {
```

```
        return 0;
    } else if (cl >= l and cr <= r) {
        return tree[pos];
    } else {
        push_down(pos, cl, cr);
        int mid = (cl + cr) / 2;
        return query(l, r, pos * 2, cl, mid) +
               query(l, r, pos * 2 + 1, mid + 1, cr);
    }
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    int q;
    cin >> n >> q;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    build();
    while (q--) {
        int op;
        cin >> op;
        if (op == 1) {
            int l, r, k;
            cin >> l >> r >> k;
            add(l, r, k);
        } else {
            int l, r;
            cin >> l >> r;
            cout << query(l, r) << '\n';
        }
    }
}
```

线段树+树剖维护路径子树和.cpp

```cpp
// 线段树+树剖维护路径子树和
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, q, root, mod, t[4 * N], mk[4 * N], top[N], hson[N], dep[N], dn[N], a[N],
    fa[N], sz[N], b[N], mxdn[N], cnt;

void build(int l, int r, int pos = 1);
void push_down(int pos, int cl, int cr);
void add(int l, int r, int x, int pos = 1, int cl = 1, int cr = n);
int query(int l, int r, int pos = 1, int cl = 1, int cr = n);
vector<int> con[N];
void dfs(int a, int f) {
    fa[a] = f;
    sz[a] = 1;
    dep[a] = dep[f] + 1;
    int mx = 0;
    for (auto t : con[a]) {
        if (t == f)
            continue;
        dfs(t, a);
        sz[a] += sz[t];
        if (sz[t] > mx) {
            mx = sz[t];
            hson[a] = t;
        }
    }
}
void df(int a) {
    dn[a] = ++cnt;
    if (hson[a]) {
        top[hson[a]] = top[a];
        df(hson[a]);
    }
    for (auto t : con[a]) {
        if (top[t])
            continue;
        top[t] = t;
        df(t);
```

```
    }
    mxdn[a] = cnt;
}
void upd_path(int a, int b, int x) {
    while (top[a] != top[b]) {
        if (dep[top[a]] > dep[top[b]]) {
            add(dn[a], dn[top[a]], x);
            a = fa[top[a]];
        } else {
            add(dn[b], dn[top[b]], x);
            b = fa[top[b]];
        }
    }
    add(dn[a], dn[b], x);
}
int query_path(int a, int b) {
    int ans = 0;
    while (top[a] != top[b]) {
        if (dep[top[a]] > dep[top[b]]) {
            ans += query(dn[a], dn[top[a]]);
            a = fa[top[a]];
        } else {
            ans += query(dn[b], dn[top[b]]);
            b = fa[top[b]];
        }
    }
    return ans + query(dn[a], dn[b]);
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> q >> root >> mod;
    for (int i = 1; i <= n; i++) {
        cin >> b[i];
    }
    for (int i = 1; i < n; i++) {
        int f, t;
        cin >> f >> t;
        con[f].push_back(t);
        con[t].push_back(f);
    }
    dfs(root, root);
    top[root] = root;
```

```
    df(root);
    for (int i = 1; i <= n; i++) {
        a[dn[i]] = b[i];
    }
    build(1, n);
    while (q--) {
        int op;
        cin >> op;
        if (op == 1) {
            int f, t, x;
            cin >> f >> t >> x;
            upd_path(f, t, x);
        } else if (op == 2) {
            int f, t;
            cin >> f >> t;
            cout << query_path(f, t) % mod << '\n';
        } else if (op == 3) {
            int s, x;
            cin >> s >> x;
            add(dn[s], mxdn[s], x);
        } else {
            int s;
            cin >> s;
            cout << query(dn[s], mxdn[s]) % mod << '\n';
        }
    }
}

void build(int l, int r, int pos) {
    if (l == r) {
        t[pos] = a[l];
    } else {
        int mid = l + r >> 1;
        build(l, mid, pos * 2), build(mid + 1, r, pos * 2 + 1);
        t[pos] = t[pos * 2] + t[pos * 2 + 1];
    }
}
void push_down(int pos, int cl, int cr) {
    int mid = cl + cr >> 1;
    mk[pos * 2] += mk[pos];
    mk[pos * 2 + 1] += mk[pos];
    t[pos * 2] += (mid - cl + 1) * mk[pos];
    t[pos * 2 + 1] += (cr - mid) * mk[pos];
```

```
    mk[pos] = 0;
}
void add(int l, int r, int x, int pos, int cl, int cr) {
    if (l > r)
        swap(l, r);
    if (cr < l or cl > r) {
        return;
    } else if (cl >= l and cr <= r) {
        t[pos] += (cr - cl + 1) * x;
        if (cr > cl)
            mk[pos] += x;
    } else {
        int mid = cl + cr >> 1;
        push_down(pos, cl, cr);
        add(l, r, x, pos * 2, cl, mid), add(l, r, x, pos * 2 + 1, mid + 1, cr);
        t[pos] = t[pos * 2] + t[pos * 2 + 1];
    }
}
int query(int l, int r, int pos, int cl, int cr) {
    if (l > r)
        swap(l, r);
    if (cr < l or cl > r) {
        return 0;
    } else if (cl >= l and cr <= r) {
        return t[pos];
    } else {
        int mid = cl + cr >> 1;
        push_down(pos, cl, cr);
        return query(l, r, pos * 2, cl, mid) +
               query(l, r, pos * 2 + 1, mid + 1, cr);
    }
}
```

长链剖分优化dp.cpp

```cpp
// 长链剖分优化 dp P7768
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 1e6 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, q, w[N], sz[N], lson[N], xo[N], dep[N];
vector<int> con[N], que[N];
int q1[N], q2[N], mxd[N];
void df(int a, int f) {
    sz[a] = 1;
    xo[a] = w[a];
    mxd[a] = dep[a] = dep[f] + 1;
    for (int t : con[a]) {
        if (t == f)
            continue;
        df(t, a);
        mxd[a] = max(mxd[a], mxd[t]);
        xo[a] ^= xo[t];
        sz[a] += sz[t];
        lson[a] = mxd[t] > mxd[lson[a]] ? t : lson[a];
    }
}
vector<int> ans[N], dp[N];
void dfs(int a, int f) {
    if (lson[a]) {
        dfs(lson[a], a);
        dp[a].swap(dp[lson[a]]);
    }
    dp[a].push_back(xo[a]);
    for (int t : con[a]) {
        if (t == f or t == lson[a])
            continue;
        dfs(t, a);
        for (int i = 0; i < dp[t].size(); i++) {
            int ia = dp[a].size() - 1 - i, it = dp[t].size() - 1 - i;
            dp[a][ia - 1] ^= dp[t][it];
        }
    }
    for (int h : que[a]) {
        if (h >= mxd[a] - dep[a]) {
```

```
            ans[a].push_back(xo[a]);
        } else {
            ans[a].push_back(xo[a] ^ dp[a][dp[a].size() - 2 - h]);
        }
    }
}
void read(int& x) {
    bool neg = false;
    x = 0;
    char ch = 0;
    while (ch < '0' || ch > '9') {
        if (ch == '-')
            neg = true;
        ch = getchar();
    }
    if (neg) {
        while (ch >= '0' && ch <= '9') {
            x = x * 10 + ('0' - ch);
            ch = getchar();
        }
    } else {
        while (ch >= '0' && ch <= '9') {
            x = x * 10 + (ch - '0');
            ch = getchar();
        }
    }
}
int pt[N];
signed main() {
    read(n);
    read(q);
    for (int i = 1; i <= n; i++) {
        read(w[i]);
    }
    for (int i = 2; i <= n; i++) {
        int f;
        read(f);
        con[f].push_back(i);
        con[i].push_back(f);
    }
    df(1, 1);
    for (int i = 1; i <= q; i++) {
        int x, h;
```

```
        read(x);
        read(h);
        q1[i] = x, q2[i] = h;
        que[x].push_back(h);
    }
    dfs(1, 1);
    for (int i = 1; i <= q; i++) {
        int x = q1[i];
        printf("%.3lf\n", ans[x][pt[x]++] * 0.001);
    }
}
```

质因数分解+记录质因数出现次数.cpp

```
// 质因数分解 记录质因数出现次数
#include <bits/stdc++.h>
using namespace std;
map<int, int> mp;
int calc(int x) {
    int x;
    cin >> x;
    for (int fac = 2; fac * fac <= x; fac++) {
        while (x % fac == 0) {
            x /= fac;
            mp[fac]++;
        }
    }
    if (x > 1) {
        mp[x]++;
    }
}
```

重链剖分求LCA.cpp

```cpp
// 重链剖分求LCA
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 5e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
vector<int> con[N];
int dep[N], hson[N], fa[N], sz[N], top[N];
void dfs(int a, int f) {
    fa[a] = f;
    sz[a] = 1;
    dep[a] = dep[f] + 1;
    int mx = 0;
    for (int t : con[a]) {
        if (t == f)
            continue;
        dfs(t, a);
        sz[a] += sz[t];
        if (sz[t] > mx) {
            mx = sz[t];
            hson[a] = t;
        }
    }
}
void df(int a) {
    for (int t : con[a]) {
        if (t == fa[a])
            continue;
        if (t == hson[a])  // top[t] = hson[a] == t ? top[a] : t;
            top[t] = top[a];
        else
            top[t] = t;
        df(t);
    }
}
int lca(int a, int b) {
    while (top[a] != top[b]) {
        if (dep[top[a]] > dep[top[b]])
            a = fa[top[a]];
        else
            b = fa[top[b]];
```

```
    }
    if (dep[a] > dep[b])
        return b;
    else
        return a;
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n, q, s;
    cin >> n >> q >> s;
    for (int i = 1; i < n; i++) {
        int f, t;
        cin >> f >> t;
        con[f].push_back(t);
        con[t].push_back(f);
    }
    dfs(s, s);
    top[s] = s;
    df(s);
    while (q--) {
        int a, b;
        cin >> a >> b;
        cout << lca(a, b) << '\n';
    }
}
```

组合数.cpp

```
// 组合数
int mod = 998244353;

int cres[67][67];
int C(int n, int m) {
    if (m == 0 || m == n)
        return 1;
    if (cres[n][m])
        return cres[n][m] % mod;
    return cres[n][m] = (C(n - 1, m) + C(n - 1, m - 1)) % mod;
}
```

最小生成树 kruskal.cpp

```cpp
// 最小生成树 kruskal
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 2e5 + 10, INF = 0x3f3f3f3f3f3f3f3f;
int n, m, fa[N];
struct Eg {
    int f, t, w;
    Eg() : f(0), t(0), w(0) {};
    Eg(int f, int t, int w) : f(f), t(t), w(w) {};
    bool operator<(const Eg& b) const { return w < b.w; }
} eg[N];
int find(int x) {
    return x == fa[x] ? x : fa[x] = find(fa[x]);
}
int kru() {
    sort(eg + 1, eg + 1 + m);
    for (int i = 1; i <= n; i++) {
        fa[i] = i;
    }
    int ans = 0, cnt = 0;
    for (int i = 1; i <= m; i++) {
        auto [f, t, w] = eg[i];
        int ff = find(f), ft = find(t);
        if (ff == ft)
            continue;
        fa[ff] = ft;
        ans += w;
        cnt++;
    }
    return cnt < n - 1 ? -1 : ans;
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        int f, t, w;
        cin >> f >> t >> w;
        eg[i] = Eg(f, t, w);
```

```
    }
    int res = kru();
    if (res == -1)
        cout << "orz";
    else
        cout << res;
}
```