

bandiaoz algorithm C++ (2022)

Contents

bandiaoz

August 26, 2022

1 DataStruct

1.1	Chtholly.cpp	1
1.2	DSU.cpp	2
1.3	LazySegmentTree.cpp	3
1.4	Mo.cpp	3
1.5	NearestPointPair.cpp	4
1.6	PointDivideAndConquer1.cpp	5
1.7	PointDivideAndConquer2.cpp	6
1.8	Segtree.cpp	7
1.9	SegtreeNoneRecursive.cpp	8
1.10	SparseTable.cpp	9
1.11	TheKthFarPointPair.cpp	9
1.12	Trie01.cpp	10
1.13	dsu_on_tree.cpp	10
1.14	fenwick.cpp	11
1.15	fhq-Treap(区间).cpp	11
1.16	fhq-Treap.cpp	12
1.17	jls线段树.cpp	13
1.18	segment_tree3.cpp	15
1.19	主席树.cpp	16
1.20	区间覆盖.cpp	16
1.21	带权并查集.cpp	17
1.22	替罪羊.cpp	18
1.23	树剖.cpp	19
1.24	笛卡尔树.cpp	21
1.25	轻重链剖分.cpp	21

2 Geometry

2.1	Circle.cpp	23
2.2	HalfPlane.cpp	23
2.3	Line.cpp	25
2.4	Point.cpp	26
2.5	PolygonAndConvex.cpp	27
2.6	Triangle.cpp	27
2.7	mygeo.cpp	29

3 Graph

3.1	2sat.cpp	36
3.2	Graph.cpp	36
3.3	MaxAssignment.cpp	36
3.4	Mincost.cpp	38
3.5	Tree.cpp	39
3.6	dijkstra.cpp	40
3.7	dinic.cpp	41
3.8	spfa.cpp	41
3.9	匈牙利.cpp	42

4 Math

4.1	China.cpp	43
4.2	Euler.cpp	43
4.3	FFT.cpp	43
4.4	Lagrange.cpp	44
4.5	Lucas.cpp	44
4.6	Miller-Rabin.cpp	45
4.7	NTT.cpp	46
4.8	basic.cpp	47
4.9	binom.cpp	47

4.10	exgcd.cpp	48
4.11	xor_basis.cpp	48
4.12	公式.md	49
4.13	区间线性基.cpp	49
4.14	取模gauss.cpp	50
4.15	容斥.cpp	50
4.16	异或gauss.cpp	51
4.17	斐波那契.cpp	52
4.18	求逆元.cpp	52
4.19	浮点型gauss.cpp	52
4.20	第二类斯特林数.cpp	53
4.21	线性基类.cpp	54
4.22	除法分块.cpp	54

5 Others

5.1	BigNum2.cpp	55
5.2	Simulated_annealing.cpp	56
5.3	Z.cpp	56
5.4	bignum.cpp	57
5.5	gen.py	59
5.6	makestd.cpp	59
5.7	pai.py	59
5.8	sg函数.cpp	59
5.9	博弈.cpp	60
5.10	威佐夫博弈.cpp	60
5.11	杜教BM.cpp	60
5.12	欧拉函数.cpp	61

6 String

6.1	AhoCorasick.cpp	61
6.2	exkmp.cpp	62
6.3	kmp.cpp	63
6.4	manacher.cpp	63
6.5	后缀数组.cpp	64

7 dp

7.1	数位dp.cpp	65
7.2	最长上升子序列.cpp	65

1 DataStruct

1.1 Chtholly.cpp

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

using namespace std;
using ll = long long;

struct Chtholly {
    struct node {
        int l, r;
        mutable ll v;

        node(int l, int r, ll v) : l(l), r(r), v(v) {}
        int size() const {
            return r - l;
        }
    };
    bool operator<(const node &A) const {
        return l < A.l;
    }
};

set<node> s;
auto insert(int l, int r, ll v) {
    return s.insert(node(l, r, v));
}
auto split(int pos) { // [l, pos), [pos, r)
```

```

    auto it = s.lower_bound(node(pos, -1, 0));
    if (it != s.end() && it->l == pos) {
        return it;
    }
    --it;
    int L = it->l, R = it->r;
    ll V = it->v;
    s.erase(it);
    insert(L, pos, V);
    //
    return insert(pos, R, V).first;
}

void add(int l, int r, ll x) { //
    for (auto itr = split(r), itl = split(l); itl != itr; ++itl) {
        itl->v += x;
    }
}

void assign_val(int l, int r, ll x) { //   x   ,   itl
    auto itr = split(r), itl = split(l); //   ,   itl
    s.erase(itl, itr);
    insert(l, r, x);
}

ll ranks(int l, int r, int k) { //   k
    vector<pair<ll, int>> vp;
    for (auto itr = split(r), itl = split(l); itl != itr; ++itl) {
        vp.push_back({itl->v, itl->size()});
    }
    sort(vp.begin(), vp.end());
    for (auto it : vp) {
        k -= it.second;
        if (k <= 0) {
            return it.first;
        }
    }
    assert(false);
    return -1;
}

ll sum(int l, int r, int ex, int mod) { //
    auto powmod = [](ll a, int b, int mod) {
        ll ans = 1;
        for (a %= mod; b; b >>= 1, a = a * a % mod) {
            if (b & 1) {
                ans = ans * a % mod;
            }
        }
        return ans;
    };

    ll res = 0;
    for (auto itr = split(r), itl = split(l); itl != itr; ++itl) {
        res = (res + itl->size() * powmod(itl->v, ex, mod)) % mod;
    }
    return res;
}

};

const int mod = 1e9 + 7;

int seed, vmax;
int rnd() {
    int ret = seed;
    seed = (seed * 7LL + 13) % mod;
    return ret;
}

int main() {
    ios::sync_with_stdio(false);

```

```

    cin.tie(nullptr);

    int n, m;
    cin >> n >> m >> seed >> vmax;

    Chtholly cho;
    for (int i = 0; i < n; ++i) {
        int x = rnd() % vmax + 1;
        cho.insert(i, i + 1, x);
    }

    while (m--) {
        int op = rnd() % 4 + 1;

        int l = rnd() % n;
        int r = rnd() % n;
        if (l > r) {
            swap(l, r);
        }
        r++;

        ll x, y;
        if (op == 3) {
            x = rnd() % (r - l) + 1;
        } else {
            x = rnd() % vmax + 1;
        }

        if (op == 4) {
            y = rnd() % vmax + 1;
        }

        if (op == 1) {
            cho.add(l, r, x);
        } else if (op == 2) {
            cho.assign_val(l, r, x);
        } else if (op == 3) {
            cout << cho.ranks(l, r, x) << "\n";
        } else {
            cout << cho.sum(l, r, x, y) << "\n";
        }
    }

    return 0;
}

```

1.2 DSU.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

struct DSU {
    vector<int> f, sz;
    DSU(int n) : f(n), sz(n, 1) { iota(f.begin(), f.end(), 0); }
    int findR(int x) { return x == f[x] ? x : f[x] = findR(f[x]); }
    bool same(int x, int y) { return findR(x) == findR(y); }
    bool merge(int x, int y) {
        x = findR(x), y = findR(y);
        if (x == y) return false;
        sz[x] += sz[y], f[y] = x;
        return true;
    }
    int size(int x) { return sz[findR(x)]; }
};

```

1.3 LazySegmentTree.cpp

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

using namespace std;
using ll = long long;

struct Info {
    ll val;
    Info(ll val = 0) : val(val) {}
    friend Info operator+(const Info &A, const Info &B) {
        return Info(A.val + B.val);
    }
};

void apply(Info &a, ll b, int l, int r) {
    a.val += b * (r - l + 1);
}

void apply(ll &a, ll b, int l, int r) {
    a += b;
}

template<class Info, class Tag, class Merge = plus<Info>>
class LazySegmentTree {
private:
    const int n;
    const Merge merge{};
    vector<Info> info; // data of segment tree, 1-index
    vector<Tag> tag; // lazy tag of segment tree

    /* [x, y] and val: Add val to each element in range of [x, y]
     * p: The id of subtree, which is an index of vector 'info'.
     * [l, r): The range of p.
     */
    void innerPull(int p) {
        info[p] = merge(info[p << 1], info[p << 1 | 1]);
    }
    void innerApply(int p, const Tag &v, int l, int r) {
        ::apply(info[p], v, l, r);
        ::apply(tag[p], v, l, r);
    }
    void push(int p, int l, int r) {
        if (tag[p] != Tag()) {
            int m = (l + r) / 2;
            innerApply(p << 1, tag[p], l, m);
            innerApply(p << 1 | 1, tag[p], m, r);
            tag[p] = Tag();
        }
    }
    void innerUpdate(int p, int x, int y, const Tag &v, int l, int r) {
        if (x <= l && r <= y) {
            innerApply(p, v, l, r);
            return;
        }
        int m = (l + r) / 2;

        push(p, l, r);
        if (x < m) innerUpdate(p << 1, x, y, v, l, m);
        if (y > m) innerUpdate(p << 1 | 1, x, y, v, m, r);
        innerPull(p);
    }
    /* Query the sum-up value of range [x, y]. */
    Info innerQuery(int p, int x, int y, int l, int r) {
        if (x <= l && r <= y) return info[p];
        if (x >= r || y <= l) return Info();
        int m = (l + r) / 2;
```

```
        push(p, l, r);
        return merge(innerQuery(p << 1, x, y, l, m), innerQuery(p << 1 | 1, x, y, m, r));
    }
};

public:
    LazySegmentTree(int n) : n(n), info(4 << (32 - __builtin_clz(n))), tag(4 << (32 - __builtin_clz(n))) {}
    LazySegmentTree(vector<Info> &init) : LazySegmentTree(init.size()) {
        function<void(int, int, int)> innerBuild = [&](int p, int l, int r) {
            if (r - l == 1) {
                info[p] = init[l];
                return;
            }
            int m = (l + r) / 2;
            innerBuild(p << 1, l, m);
            innerBuild(p << 1 | 1, m, r);
            innerPull(p);
        };
        innerBuild(1, 0, n);
    }
    /* Add val to each element in range of [x, y] */
    void update(int x, int y, Tag v) {
        innerUpdate(1, x, y, v, 0, n);
    }
    /* Query the sum-up value of range [x, y] */
    Info query(int x, int y) {
        return innerQuery(1, x, y, 0, n);
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<Info> a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i].val;
    }

    LazySegmentTree<Info, ll> seg(a);
    for (int i = 0; i < m; ++i) {
        ll op, x, y, k;
        cin >> op >> x >> y;
        x--;
        if (op == 1) {
            cin >> k;
            seg.update(x, y, k);
        } else if (op == 2) {
            cout << seg.query(x, y).val << "\n";
        }
    }

    return 0;
}

// test problem: https://www.luogu.com.cn/problem/P3372
```

1.4 Mo.cpp

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

using namespace std;
using ll = long long;
```

```

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
        a[i]--;
    }

    int q;
    cin >> q;
    vector<int> l(q), r(q);
    for (int i = 0; i < q; ++i) {
        cin >> l[i] >> r[i];
        l[i]--;
    }

    const int B = max(1.0, n / sqrt(q));
    vector<int> p(q);
    iota(p.begin(), p.end(), 0);
    sort(p.begin(), p.end(), [&](int i, int j) {
        if (l[i] / B == l[j] / B) return r[i] < r[j];
        else return l[i] < l[j];
    });

    vector<int> cnt(n);
    int L = 0, R = 0, res = 0;
    auto add = [&](int x, int f) {
        res -= cnt[x] / 2;
        cnt[x] += f;
        res += cnt[x] / 2;
    };

    vector<int> ans(q);
    for (auto i : p) {
        while (L > l[i]) add(a[--L], 1);
        while (R < r[i]) add(a[R++], 1);
        while (L < l[i]) add(a[L++], -1);
        while (R > r[i]) add(a[R--], -1);
        ans[i] = res;
    }

    for (int i = 0; i < q; ++i) {
        cout << ans[i] << "\n";
    }

    return 0;
}

```

// https://atcoder.jp/contests/abc242/tasks/abc242_g

1.5 NearestPointPair.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

template<typename T, int K = 2>
struct KDTree {
    KDTree(int n) : n(n), lc(n, -1), rc(n, -1), boundary(n, vector<vector<T>>(K, vector<T>(2))) {}
    KDTree(vector<array<T, K>> &st) : KDTree(st.size()) {}

```

```

    a = st;
    function<int(int, int, int)> innerBuild = [&](int l, int r, int div) {
        if (l >= r) {
            return -1;
        }
        int mid = (l + r) >> 1;
        nth_element(a.begin() + l, a.begin() + mid, a.begin() + r, Cmp(div));
        lc[mid] = innerBuild(l, mid, (div + 1) % K);
        rc[mid] = innerBuild(mid + 1, r, (div + 1) % K);
        maintain(mid);
        return mid;
    };

    innerBuild(0, n, 0);
};

void query(int p, T &ans) {
    innerQuery(0, n, p, ans);
}

private:
    const int n;
    vector<int> lc, rc;
    vector<vector<vector<T>>> boundary;
    vector<array<T, K>> a;

    struct Cmp {
        int div;
        Cmp(const int &div) : div(div) {}
        bool operator()(const array<T, K> &A, const array<T, K> &B) {
            for (int i = 0; i < K; ++i) {
                if (A[(i + div) % K] != B[(i + div) % K]) {
                    return A[(i + div) % K] < B[(i + div) % K];
                }
            }
            return false;
        }
    };

    bool cmp(const array<T, K> &A, const array<T, K> &B, int div) {
        Cmp cp(div);
        return cp(A, B);
    }

    template<typename U> U sqr(U x) { return x * x; }
    T dis(const array<T, K> &A, const array<T, K> &B) {
        T ans = 0;
        for (int i = 0; i < K; ++i) {
            ans += sqr(A[i] - B[i]);
        }
        return ans;
    }

    void maintain(int i) {
        for (int j = 0; j < K; ++j) {
            boundary[i][j][0] = boundary[i][j][1] = a[i][j];
            if (lc[i] != -1) {
                boundary[i][j][0] = min(boundary[i][j][0], boundary[lc[i]][j][0]);
                boundary[i][j][1] = max(boundary[i][j][1], boundary[lc[i]][j][1]);
            }
            if (rc[i] != -1) {
                boundary[i][j][0] = min(boundary[i][j][0], boundary[rc[i]][j][0]);
                boundary[i][j][1] = max(boundary[i][j][1], boundary[rc[i]][j][1]);
            }
        }
    }

    T fmin(int p, int i) { // the minimum distance to this area
        // if i == -1, ignore this area when calculating the answer.
        if (i == -1) {
            return 1e18;
        }
        T ans = 0;

```

```

    for (int j = 0; j < K; ++j) {
        if (a[p][j] < boundary[i][j][0]) ans += sqr(boundary[i][j][0] - a[p][j]);
        if (a[p][j] > boundary[i][j][1]) ans += sqr(a[p][j] - boundary[i][j][1]);
    }
    return ans;
}

void innerQuery(int l, int r, int p, T &ans) {
    if (l >= r) return;
    int mid = (l + r) >> 1;
    if (p != mid) {
        ans = min(ans, dis(a[p], a[mid]));
    }
    if (l + 1 == r) return;

    T dl = fmin(p, lc[mid]), dr = fmin(p, rc[mid]);
    if (dl < ans && dr < ans) {
        if (dl < dr) {
            innerQuery(l, mid, p, ans);
            if (dr < ans) {
                innerQuery(mid + 1, r, p, ans);
            }
        } else {
            innerQuery(mid + 1, r, p, ans);
            if (dl < ans) {
                innerQuery(l, mid, p, ans);
            }
        }
    } else if (dl < ans) {
        innerQuery(l, mid, p, ans);
    } else if (dr < ans) {
        innerQuery(mid + 1, r, p, ans);
    }
}

}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<array<double, 2>> a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i][0] >> a[i][1];
    }

    KDTree<double> kdt(a);

    double ans = 2e18;
    for (int i = 0; i < n; ++i) {
        kdt.query(i, ans);
    }

    cout << fixed << setprecision(4) << sqrt(ans) << "\n";

    return 0;
}

```

// test problem: <https://www.luogu.com.cn/problem/P1429>

1.6 PointDivideAndConquer1.cpp

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

```
using namespace std;
using ll = long long;
```

```

template <typename T>
struct Fenwick {
    const int n;
    vector<T> a;
    Fenwick(int n) : n(n), a(n) {}
    void add(int x, T v) {
        for (int i = x + 1; i <= n; i += i & -i) {
            a[i - 1] += v;
        }
    }
    // return the sum of [0, x)
    T sum(int x) {
        T ans = 0;
        for (int i = x; i > 0; i -= i & -i) {
            ans += a[i - 1];
        }
        return ans;
    }
    // return the sum of [l, r)
    T rangeSum(int l, int r) {
        return sum(r) - sum(l);
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;
    vector<vector<pair<int, int>>> g(n);
    vector<int> w(n - 1);
    for (int i = 0; i < n - 1; ++i) {
        int u, v;
        cin >> u >> v >> w[i];
        u--, v--;
        g[u].emplace_back(v, i);
        g[v].emplace_back(u, i);
    }

    int k;
    cin >> k;

    vector<int> sz(n);
    vector<bool> vis(n);
    Fenwick<int> fen(k + 1);
    function<void(int, int, int, int)> dfs_rt = [&](int u, int f, int tot, int &rt)
    {
        int maxx = 0;
        sz[u] = 1;
        for (auto [v, j] : g[u]) {
            if (v == f || vis[v]) continue;
            dfs_rt(v, u, tot, rt);
            sz[u] += sz[v];
            maxx = max(maxx, sz[v]);
        }
        maxx = max(maxx, tot - sz[u]);
        if (maxx * 2 <= tot) {
            rt = u;
        }
    };

    function<void(int, int)> dfs_sz = [&](int u, int f) {
        sz[u] = 1;
        for (auto [v, j] : g[u]) {
            if (v == f || vis[v]) continue;

```

```

        dfs_sz(v, u);
        sz[u] += sz[v];
    }
};

vector<int> d;
function<void(int, int, int)> dfs_dis = [&](int u, int f, int dis) {
    d.push_back(dis);
    for (auto [v, j] : g[u]) {
        if (v == f || vis[v]) continue;
        dfs_dis(v, u, dis + w[j]);
    }
};

function<void(int, int, int)> dfs_clear = [&](int u, int f, int dis) {
    if (dis) fen.add(dis, -1);
    for (auto [v, j] : g[u]) {
        if (v == f || vis[v]) continue;
        dfs_clear(v, u, dis + w[j]);
    }
};

function<int(int, int)> work = [&](int u, int tot) {
    int rt = u;
    dfs_rt(u, -1, tot, rt);
    dfs_sz(rt, -1);
    vis[rt] = true;

    int ans = 0;
    for (auto [v, j] : g[rt]) {
        if (vis[v]) continue;
        d.clear();
        dfs_dis(v, rt, w[j]);
        for (auto dd : d) {
            if (dd <= k) {
                ans += fen.sum(k - dd + 1) + 1;
            }
        }
        for (auto dd : d) {
            fen.add(dd, 1);
        }
    }
    dfs_clear(rt, -1, 0);
    for (auto [v, j] : g[rt]) {
        if (vis[v]) continue;
        ans += work(v, sz[v]);
    }
    return ans;
};

cout << work(0, n) << "\n";

return 0;
}

```

// test problem: <https://www.luogu.com.cn/problem/P4178>

1.7 PointDivideAndConquer2.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

```

```

    int n, m;
    cin >> n >> m;
    vector<vector<pair<int, int>>> g(n);
    vector<int> w(n);
    for (int i = 0; i < n - 1; ++i) {
        int u, v;
        cin >> u >> v >> w[i];
        u--, v--;
        g[u].emplace_back(v, i);
        g[v].emplace_back(u, i);
    }

    vector<int> ans(m), Q(m);
    for (int i = 0; i < m; ++i) {
        cin >> Q[i];
    }

    vector<int> sz(n);
    vector<bool> vis(n);
    function<void(int, int, int, int)> dfs_rt = [&](int u, int f, int tot, int &rt)
    {
        int maxx = 0;
        sz[u] = 1;
        for (auto [v, j] : g[u]) {
            if (v == f || vis[v]) continue;
            dfs_rt(v, u, tot, rt);
            sz[u] += sz[v];
            maxx = max(maxx, sz[v]);
        }
        maxx = max(maxx, tot - sz[u]);
        if (maxx * 2 <= tot) {
            rt = u;
        }
    };

    function<void(int, int)> dfs_sz = [&](int u, int f) {
        sz[u] = 1;
        for (auto [v, j] : g[u]) {
            if (v == f || vis[v]) continue;
            dfs_sz(v, u);
            sz[u] += sz[v];
        }
    };

    vector<bool> mpd(10000001);
    int cnt;
    vector<int> d(n);

    function<void(int, int, int)> dfs_ans = [&](int u, int f, int dis) {
        ++cnt;
        d[u] = dis;
        for (int i = 0; i < m; ++i) {
            if (d[u] == Q[i]) {
                ans[i] = true;
            } else if (d[u] < Q[i]) {
                ans[i] |= mpd[Q[i] - d[u]];
            }
        }
        for (auto [v, j] : g[u]) {
            if (v == f || vis[v]) continue;
            dfs_ans(v, u, dis + w[j]);
        }
    };

    function<void(int, int, int)> dfs_dis = [&](int u, int f, int flag) {

```

```

    for (int i = 0; i < m; ++i) {
        if (d[u] <= Q[i]) {
            mpd[d[u]] = (flag == 1);
        }
    }
    for (auto [v, j] : g[u]) {
        if (v == f || vis[v]) continue;
        dfs_dis(v, u, flag);
    }
}
};

```

```

function<void(int, int)> work = [&](int u, int tot) {
    int rt = u;
    dfs_rt(u, -1, tot, rt);
    dfs_sz(rt, -1);
    vis[rt] = true;

```

```

    for (auto [v, j] : g[rt]) {
        if (vis[v]) continue;
        dfs_ans(v, rt, w[j]);
        dfs_dis(v, rt, 1);
    }

```

```

    dfs_dis(rt, -1, -1);

```

```

    for (auto [v, j] : g[rt]) {
        if (vis[v]) continue;
        work(v, sz[v]);
    }

```

```

};

```

```

work(0, n);

```

```

for (int i = 0; i < m; ++i) {
    cout << (ans[i] ? "AYE" : "NAY") << "\n";
}

```

```

return 0;

```

```

}

```

1.8 Segtree.cpp

```

#include <bits/stdc++.h>

```

```

using namespace std;
using ll = long long;

```

```

template<class Info, class Merge = plus<Info>>
struct SegmentTree {
    SegmentTree(int n) : n(n), merge(Merge()), info(4 << (32 - __builtin_clz(n))) {}
    SegmentTree(vector<Info> init) : SegmentTree(init.size()) {
        function<void(int, int, int)> build = [&](int p, int l, int r) {
            if (r - l == 1) {
                info[p] = init[l];
                return;
            }
            int mid = (l + r) / 2;
            build(p << 1, l, mid);
            build(p << 1 | 1, mid, r);
            innerPull(p);
        };
        build(1, 0, n);
    }
    void modify(int pos, const Info &x) {
        innerModify(1, 0, n, pos, x);
    }

```

```

    }
    Info rangeQuery(int l, int r) {
        return innerRangeQuery(1, 0, n, l, r);
    }

```

```

private:

```

```

    const int n;
    const Merge merge;
    vector<Info> info;
    void innerPull(int p) {
        info[p] = merge(info[p << 1], info[p << 1 | 1]);
    }
    void innerModify(int p, int l, int r, int pos, const Info &x) {
        if (r - l == 1) {
            info[p] = info[p] + x;
            return;
        }
        int mid = (l + r) / 2;
        if (pos < mid) {
            innerModify(p << 1, l, mid, pos, x);
        } else {
            innerModify(p << 1 | 1, mid, r, pos, x);
        }
        innerPull(p);
    }
    Info innerRangeQuery(int p, int l, int r, int x, int y) {
        if (l >= y || r <= x) return Info();
        if (l >= x && r <= y) return info[p];
        int mid = (l + r) / 2;
        return merge(innerRangeQuery(p << 1, l, mid, x, y), innerRangeQuery(p << 1 | 1, mid, r, x, y));
    }

```

```

};

```

```

struct Info {
    int val;
    Info(int val = 0) : val(val) {}
    friend Info operator+(const Info &A, const Info &B) {
        return Info(A.val + B.val);
    }
};

```

```

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;
    SegmentTree<Info> seg(n);
    for (int i = 0; i < n; ++i) {
        int x;
        cin >> x;
        seg.modify(i, x);
    }

    while (m--) {
        int op, x, y;
        cin >> op;
        if (op == 1) {
            cin >> x >> y;
            x--;
            seg.modify(x, y);
        } else {
            cin >> x >> y;
            x--;
            cout << seg.rangeQuery(x, y).val << "\n";
        }
    }
}

```

```

    }
    return 0;
}

```

// test problem: <https://www.luogu.com.cn/problem/P3374>

1.9 SegtreeNoneRecursive.cpp

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

```
using namespace std;
using ll = long long;
```

```
constexpr unsigned ceil_lg(int n) {
    return n == 0 ? 0 : 32 - __builtin_clz(n - 1);
}
template <typename T> struct Segtree {
public:
    Segtree() : Segtree(0) {}
    explicit Segtree(int n) : Segtree(vector<typename T::S>(n, T::e())) {}
    explicit Segtree(const vector<typename T::S>& a) : _n(int(a.size())) {
        log = ceil_lg(_n);
        size = 1 << log;
        d = vector<typename T::S>(2 * size, T::e());
        for (int i = 0; i < _n; i++) d[size + i] = a[i];
        for (int i = size - 1; i >= 1; i--) {
            update(i);
        }
    }
    void set(int p, typename T::S x) {
        assert(0 <= p && p < _n);
        p += size;
        d[p] = x;
        for (int i = 1; i <= log; i++) update(p >> i);
    }
    typename T::S get(int p) const {
        assert(0 <= p && p < _n);
        return d[p + size];
    }
    typename T::S query(int l, int r) const {
        assert(0 <= l && l <= r && r <= _n);
        typename T::S sml = T::e(), smr = T::e();
        l += size;
        r += size;
        while (l < r) {
            if (l & 1) sml = T::op(sml, d[l++]);
            if (r & 1) smr = T::op(d[--r], smr);
            l >>= 1;
            r >>= 1;
        }
        return T::op(sml, smr);
    }
    typename T::S queryAll() const { return d[1]; }
    template <bool (*f)(typename T::S)> int max_right(int l) const {
        return max_right(l, [](typename T::S x) { return f(x); });
    }
    // r = l or f(op(a[l], ..., a[r - 1])) = true
    // r = n or f(op(a[l], ..., a[r])) = false
    template <class F> int max_right(int l, F f) const {
        assert(0 <= l && l <= _n);
        assert(f(T::e()));
        if (l == _n) return _n;
        l += size;
        typename T::S sm = T::e();
        do {
            while (l % 2 == 0) l >>= 1;

```

```

            if (!f(T::op(sm, d[l]))) {
                while (l < size) {
                    l = (2 * l);
                    if (f(T::op(sm, d[l]))) {
                        sm = T::op(sm, d[l]);
                        l++;
                    }
                }
                return l - size;
            }
            sm = T::op(sm, d[l]);
            l++;
        } while ((l & -l) != 1);
        return _n;
    }
    template <bool (*f)(typename T::S)> int min_left(int r) const {
        return min_left(r, [](typename T::S x) { return f(x); });
    }
    // r = l or f(op(a[l], ..., a[r - 1])) = true
    // r = n or f(op(a[l - 1], ..., a[r - 1])) = false
    template <class F> int min_left(int r, F f) const {
        assert(0 <= r && r <= _n);
        assert(f(T::e()));
        if (r == 0) return 0;
        r += size;
        typename T::S sm = T::e();
        do {
            r--;
            while (r > 1 && (r % 2)) r >>= 1;
            if (!f(T::op(d[r], sm))) {
                while (r < size) {
                    r = (2 * r + 1);
                    if (f(T::op(d[r], sm))) {
                        sm = T::op(d[r], sm);
                        r--;
                    }
                }
                return r + 1 - size;
            }
            sm = T::op(d[r], sm);
        } while ((r & -r) != r);
        return 0;
    }
private:
    int _n, size, log;
    vector<typename T::S> d;
    void update(int k) { d[k] = T::op(d[2 * k], d[2 * k + 1]); }
};

struct SegtreeOP {
    using S = int;
    static S e() { return -1; }
    static S op(const S &x, const S &y) {
        return max(x, y);
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;
    vector<int> a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
    }

```



```

Segtree<SegtreeOP> seg(a);
for (int i = 0; i < m; ++i) {
    int op;
    cin >> op;

    if (op == 1) {
        int x, v;
        cin >> x >> v;
        x--;
        seg.set(x, v);
    } else if (op == 2) {
        int l, r;
        cin >> l >> r;
        l--;
        cout << seg.query(l, r) << "\n";
    } else {
        int x, v;
        cin >> x >> v;
        x--;
        cout << seg.max_right(x, [&](int a) { return a < v; }) + 1 << "\n";
    }
}

return 0;
}

// test problem: https://atcoder.jp/contests/practice2/tasks/practice2_j
// reference: https://atcoder.github.io/ac-library/master/document_en/segtree.html

```

1.10 SparseTable.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

// usage:
// auto fun = [&](int i, int j) { return min(i, j); };
// SparseTable<int, decltype(fun)> st(a, fun);
// or:
// SparseTable<int> st(a, [&](int i, int j) { return min(i, j); });
// __builtin_clz() : Calculate the number of leading zeros

template <typename T, class F = function<T(const T&, const T&)>>
struct SparseTable {
    int n;
    vector<vector<T>> mat;
    F func;

    SparseTable(const vector<T>& a, const F& f) : func(f) {
        n = static_cast<int>(a.size());
        int max_log = 32 - __builtin_clz(n);
        mat.resize(max_log);
        mat[0] = a;
        for (int j = 1; j < max_log; j++) {
            mat[j].resize(n - (1 << j) + 1);
            for (int i = 0; i <= n - (1 << j); i++) {
                mat[j][i] = func(mat[j - 1][i], mat[j - 1][i + (1 << (j - 1))]);
            }
        }
    }

    // return the answer [from, to)
    T get(int from, int to) const {
        assert(0 <= from && from <= to && to <= n);
        int lg = 32 - __builtin_clz(to - from) - 1;

```

```

        return func(mat[lg][from], mat[lg][to - (1 << lg)]);
    }
};

```

1.11 TheKthFarPointPair.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

template<typename T, int K = 2>
struct KDTree {
    KDTree(int n) : n(n), lc(n, -1), rc(n, -1), boundary(n, vector<vector<T>>(K, vector<T>(2))) {}
    KDTree(vector<array<T, K>> &st) : KDTree(st.size()) {
        a = st;
        function<int(int, int, int)> innerBuild = [&](int l, int r, int div) {
            if (l >= r) {
                return -1;
            }
            int mid = (l + r) >> 1;
            nth_element(a.begin() + l, a.begin() + mid, a.begin() + r, Cmp(div));
            lc[mid] = innerBuild(l, mid, (div + 1) % K);
            rc[mid] = innerBuild(mid + 1, r, (div + 1) % K);
            maintain(mid);
            return mid;
        };

        innerBuild(0, n, 0);
    };

    T query(int k) {
        priority_queue<T, vector<T>, greater<T>> q;
        for (int i = 0; i < k; ++i) q.push(0);
        for (int i = 0; i < n; ++i) {
            innerQuery(0, n, i, q);
        }
        return q.top();
    }

private:
    const int n;
    vector<int> lc, rc;
    vector<vector<vector<T>>> boundary;
    vector<array<T, K>> a;

    struct Cmp {
        int div;
        Cmp(const int &div) : div(div) {}
        bool operator()(const array<T, K> &A, const array<T, K> &B) {
            for (int i = 0; i < K; ++i) {
                if (A[(i + div) % K] != B[(i + div) % K]) {
                    return A[(i + div) % K] < B[(i + div) % K];
                }
            }
            return false;
        }
    };

    bool cmp(const array<T, K> &A, const array<T, K> &B, int div) {
        Cmp cp(div);
        return cp(A, B);
    }

    template<typename U> U sqr(U x) { return x * x; }
    T dis(const array<T, K> &A, const array<T, K> &B) {
        T ans = 0;
        for (int i = 0; i < K; ++i) {
            ans += sqr(A[i] - B[i]);
        }
    }
};

```

```

        return ans;
    }
    void maintain(int i) {
        for (int j = 0; j < K; ++j) {
            boundary[i][j][0] = boundary[i][j][1] = a[i][j];
            if (lc[i] != -1) {
                boundary[i][j][0] = min(boundary[i][j][0], boundary[lc[i]][j][0]);
                boundary[i][j][1] = max(boundary[i][j][1], boundary[lc[i]][j][1]);
            }
            if (rc[i] != -1) {
                boundary[i][j][0] = min(boundary[i][j][0], boundary[rc[i]][j][0]);
                boundary[i][j][1] = max(boundary[i][j][1], boundary[rc[i]][j][1]);
            }
        }
    }
    T fmax(int p, int i) { // the maximum distance to this area
        // if i == -1, ignore this area when calculating the answer.
        if (i == -1) {
            return 0;
        }
        T ans = 0;
        for (int j = 0; j < K; ++j) {
            ans += max(sqrt(a[p][j] - boundary[i][j][0]), sqrt(a[p][j] - boundary[i][j][1]));
        }
        return ans;
    }
    void innerQuery(int l, int r, int p, priority_queue<T, vector<T>, greater<T>> &q)
    {
        if (l >= r) return;
        int mid = (l + r) >> 1;
        T tmp = dis(a[p], a[mid]);
        if (tmp > q.top()) {
            q.pop();
            q.push(tmp);
        }
        T dl = fmax(p, lc[mid]), dr = fmax(p, rc[mid]);
        if (dl > q.top() && dr > q.top()) {
            if (dl > dr) {
                innerQuery(l, mid, p, q);
                if (dr > q.top()) {
                    innerQuery(mid + 1, r, p, q);
                }
            } else {
                innerQuery(mid + 1, r, p, q);
                if (dl > q.top()) {
                    innerQuery(l, mid, p, q);
                }
            }
        } else if (dl > q.top()) {
            innerQuery(l, mid, p, q);
        } else if (dr > q.top()) {
            innerQuery(mid + 1, r, p, q);
        }
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

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

    k *= 2;

    vector<array<ll, 2>> a(n);

```

```

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

    KDTree<ll> kdt(a);

    cout << kdt.query(k) << "\n";

    return 0;
}

// test problem: https://www.luogu.com.cn/problem/P4357

```

1.12 Trie01.cpp

```

// 01 Trie find maximal xor sum
template <typename T, int B = 30>
class Trie01 {
    using Node = array<int, 2>;
    vector<Node> ch_;
    void addNode(int fa, int c) {
        ch_[fa][c] = ch_.size();
        ch_.emplace_back(Node());
    }

public:
    Trie01() : ch_(1) {}
    void insert(T x) {
        for (int i = B, p = 0; i >= 0; --i) {
            int c = x >> i & 1;
            if (ch_[p][c] == 0) addNode(p, c);
            p = ch_[p][c];
        }
    }
    T getMax(T x) {
        T res = 0;
        for (int i = B, p = 0; i >= 0; --i) {
            int c = x >> i & 1;
            if (ch_[p][c ^ 1]) {
                p = ch_[p][c ^ 1];
                res |= 1 << i;
            } else {
                p = ch_[p][c];
            }
        }
        return res;
    }
    T getMin(T x) {
        T res = 0;
        for (int i = B, p = 0; i >= 0; --i) {
            int c = x >> i & 1;
            if (ch_[p][c]) {
                p = ch_[p][c];
            } else {
                p = ch_[p][c ^ 1];
                res |= 1 << i;
            }
        }
        return res;
    }
};

```

1.13 dsu_on_tree.cpp

```

#include <bits/stdc++.h>

using namespace std;

```

```

using ll = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;
    vector<int> a(n);
    vector<vector<int>> g(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
    }
    for (int i = 0; i < n - 1; ++i) {
        int u, v;
        cin >> u >> v;
        u--, v--;
        g[u].push_back(v);
        g[v].push_back(u);
    }

    vector<int> fa(n, -1), sz(n, 1);
    function<void(int)> dfs_son = [&](int u) {
        if (u > 0) {
            g[u].erase(find(g[u].begin(), g[u].end(), fa[u]));
        }
        for (auto &v : g[u]) {
            fa[v] = u;
            dfs_son(v);
            sz[u] += sz[v];
            if (sz[v] > sz[g[u][0]]) {
                swap(v, g[u][0]);
            }
        }
    };

    dfs_son(0);

    int flag = -1, maxx = 0;
    vector<int> cnt(n + 1);
    vector<ll> ans(n);
    ll sum = 0;
    function<void(int, int)> count = [&](int u, int val) {
        cnt[a[u]] += val;
        if (cnt[a[u]] > maxx) {
            maxx = cnt[a[u]];
            sum = a[u];
        } else if (cnt[a[u]] == maxx) {
            sum += a[u];
        }
        for (auto v : g[u]) {
            if (v == flag) continue;
            count(v, val);
        }
    };

    function<void(int, bool)> dfs_dsu = [&](int u, bool keep) {
        //
        for (auto v : g[u]) {
            if (v == g[u][0]) continue;
            dfs_dsu(v, 0);
        }
        //
        if (g[u].size()) {
            dfs_dsu(g[u][0], true);
            flag = g[u][0];
        }
    };
}

```

```

//
count(u, 1);
flag = -1;
ans[u] = sum;
//
if (!keep) {
    count(u, -1);
    sum = maxx = 0;
}
};

dfs_dsu(0, false);

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

return 0;
}

// https://codeforces.com/problemset/problem/600/E

```

1.14 fenwick.cpp

```

template <typename T>
struct Fenwick {
    const int n;
    vector<T> a;
    Fenwick(int n) : n(n), a(n) {}
    void add(int x, T v) {
        for (int i = x + 1; i <= n; i += i & -i) {
            a[i - 1] += v;
        }
    }
    // return the sum of [0, x)
    T sum(int x) {
        T ans = 0;
        for (int i = x; i > 0; i -= i & -i) {
            ans += a[i - 1];
        }
        return ans;
    }
    // return the sum of [l, r)
    T rangeSum(int l, int r) {
        return sum(r) - sum(l);
    }
};

```

1.15 fhq-Treap(区间).cpp

```

#include <bits/stdc++.h>
#define rep(i, a, n) for (int i = a; i <= n; ++i)
#define per(i, a, n) for (int i = n; i >= a; --i)
#ifdef LOCAL
#include "Print.h"
#define de(...) W(' ', #__VA_ARGS__, " ") =, __VA_ARGS__
#else
#define de(...)
#endif
using namespace std;
typedef long long ll;
const int maxn = 1e5 + 5;
namespace fhq {
#define tr t[root]
#define lson t[tr.lc]
#define rson t[tr.rc]
mt19937 rnd(233);

```

```

struct node {
    int lc, rc, val, key, sz;
    bool tag;
} t[maxn];
int cnt, Root;
//    root
inline void update(int root) { tr.sz = lson.sz + rson.sz + 1; }
//    val
int newNode(int val) {
    t[++cnt] = {0, 0, val, (int)rnd(), 1, 0};
    return cnt;
}
inline void pushdown(int root) {
    swap(tr.lc, tr.rc);
    lson.tag ^= 1, rson.tag ^= 1;
    tr.tag = false;
}
//    xy
int merge(int x, int y) {
    if (!x || !y) return x + y;
    if (t[x].key < t[y].key) {
        if (t[x].tag) pushdown(x);
        t[x].rc = merge(t[x].rc, y);
        update(x); return x;
    } else {
        if (t[y].tag) pushdown(y);
        t[y].lc = merge(x, t[y].lc);
        update(y); return y;
    }
}
//    root    xk
void split_sz(int root, int k, int &x, int &y) {
    if (!root) x = y = 0;
    else {
        if (tr.tag) pushdown(root);
        if (k <= lson.sz) y = root, split_sz(tr.lc, k, x, tr.lc);
        else x = root, split_sz(tr.rc, k - lson.sz - 1, tr.rc, y);
        update(root);
    }
}
void reverse(int l, int r) {
    int x, y, z;
    split_sz(Root, l - 1, x, y);
    split_sz(y, r - l + 1, y, z);
    t[y].tag ^= 1;
    Root = merge(merge(x, y), z);
}
void ldr(int root) {
    if (!root) return;
    if (tr.tag) pushdown(root);
    ldr(tr.lc);
    printf("%d ", tr.val);
    ldr(tr.rc);
}
#undef tr
#undef lson
#undef rson
} // namespace fhq
int case_Test() {
    int n, m;
    scanf("%d%d", &n, &m);
    rep(i, 1, n) fhq::Root = fhq::merge(fhq::Root, fhq::newNode(i));
    while (m--) {
        int l, r;
        scanf("%d%d", &l, &r);
        fhq::reverse(l, r);
    }
}

```

```

fhq::ldr(fhq::Root);
return 0;
}
int main() {
#ifdef LOCAL
    freopen("/Users/chenjinglong/Desktop/cpp_code/in.in", "r", stdin);
    freopen("/Users/chenjinglong/Desktop/cpp_code/out.out", "w", stdout);
    clock_t start = clock();
#endif
    int _ = 1;
    // scanf("%d", &_);
    while (_--) case_Test();
#ifdef LOCAL
    printf("Time used: %.3lfs\n", (double)(clock() - start) / CLOCKS_PER_SEC);
#endif
    return 0;
}

```

1.16 fhq-Treap.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

template<typename key_t>
struct Treap {
    struct Node {
        key_t key;
        int pri;
        int l, r, sz;
        Node(key_t a, int b) : key(a), pri(b), l(-1), r(-1), sz(1) {}
    };

    int root = -1;
    vector<Node> tree;

    // split by key, the key of x treap less than y treap
    array<int, 2> split(int pos, key_t key) {
        if (pos == -1) return {-1, -1};

        if (tree[pos].key <= key) {
            array<int, 2> res = split(tree[pos].r, key);
            tree[pos].r = res[0];
            update(pos);
            return {pos, res[1]};
        } else {
            array<int, 2> res = split(tree[pos].l, key);
            tree[pos].l = res[1];
            update(pos);
            return {res[0], pos};
        }
    }

    // split by size, the size of x treap equal to sz
    array<int, 2> split_sz(int pos, int sz) {
        if (pos == -1) return {-1, -1};

        if (tree[tree[pos].l].sz + 1 <= sz) {
            array<int, 2> res = split_sz(tree[pos].r, sz - tree[tree[pos].l].sz - 1);
            tree[pos].r = res[0];
            update(pos);
            return {pos, res[1]};
        } else {
            array<int, 2> res = split_sz(tree[pos].l, sz);
            tree[pos].l = res[1];
            update(pos);
            return {res[0], pos};
        }
    }
}

```

```

    }
}
// small root heap, the key of x treap less than y treap
int merge(int x, int y) {
    if (x == -1) return y;
    if (y == -1) return x;

    if (tree[x].pri > tree[y].pri) {
        swap(x, y);
    }

    array<int, 2> res = split(y, tree[x].key);
    tree[x].l = merge(tree[x].l, res[0]);
    tree[x].r = merge(tree[x].r, res[1]);
    update(x);
    return x;
}

void update(int pos) {
    tree[pos].sz = tree[tree[pos].l].sz + tree[tree[pos].r].sz + 1;
}

int create(key_t key) {
    mt19937 rng((unsigned int) chrono::steady_clock::now().time_since_epoch().
        count());
    int pri = (int)(rng() & ((1ll << 31) - 1));
    tree.emplace_back(key, pri);
    return (int)tree.size() - 1;
}

void insert(int &pos, key_t key) {
    int o = create(key);
    array<int, 2> res = split(pos, key);
    pos = merge(merge(res[0], o), res[1]);
}

// Return rank with power is key
int rank(int &pos, key_t key) {
    array<int, 2> res = split(pos, key - 1);
    int rk = (res[0] == -1) ? 1 : tree[res[0]].sz + 1;
    pos = merge(res[0], res[1]);
    return rk;
}

// Return the key of the k largest
key_t kth(int &pos, int k) {
    assert(k <= tree[pos].sz);
    array<int, 2> res1 = split_sz(pos, k);
    array<int, 2> res2 = split_sz(res1[0], k - 1);
    key_t key = tree[res2[1]].key;
    pos = merge(merge(res2[0], res2[1]), res1[1]);
    return key;
}

// Delete one node that equal to key
void erase(int &pos, key_t key) {
    array<int, 2> res1 = split(pos, key);
    array<int, 2> res2 = split(res1[0], key - 1);

    if (res2[1] != -1) {
        res2[1] = merge(tree[res2[1]].l, tree[res2[1]].r);
    }

    pos = merge(merge(res2[0], res2[1]), res1[1]);
}

// Return the precursor of key
key_t pre(int &pos, key_t key) {
    array<int, 2> res = split(pos, key - 1);
    key_t ans = kth(res[0], tree[res[0]].sz);
    pos = merge(res[0], res[1]);
    return ans;
}

// Return the next of key

```

```

key_t nxt(int &pos, key_t key) {
    array<int, 2> res = split(pos, key);
    int ans = kth(res[1], 1);
    pos = merge(res[0], res[1]);
    return ans;
}

void insert(key_t x) { insert(root, x); }
void erase(int x) { erase(root, x); }
int rank(key_t x) { return rank(root, x); }
key_t kth(int x) { return kth(root, x); }
key_t pre(key_t x) { return pre(root, x); }
key_t nxt(key_t x) { return nxt(root, x); }

};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    Treap<int> T;

    for (int i = 1; i <= n; i++) {
        int op, x;
        cin >> op >> x;

        if (op == 1) {
            T.insert(x);
        } else if (op == 2) {
            T.erase(x);
        } else if (op == 3) {
            cout << T.rank(x) << "\n";
        } else if (op == 4) {
            cout << T.kth(x) << "\n";
        } else if (op == 5) {
            cout << T.pre(x) << "\n";
        } else if (op == 6) {
            cout << T.nxt(x) << "\n";
        }
    }

    return 0;
}

```

// test problem: <https://loj.ac/p/104>

1.17 jls线段树.cpp

```

#pragma region
#include <algorithm>
#include <cmath>
#include <cstring>
#include <iomanip>
#include <iostream>
#include <map>
#include <queue>
#include <set>
#include <stack>
#include <string>
#include <vector>
using namespace std;
typedef long long ll;
#define tr t[root]
#define lson t[root << 1]
#define rson t[root << 1 | 1]

```

```

#define rep(i, a, n) for (int i = a; i <= n; ++i)
#define per(i, a, n) for (int i = n; i >= a; --i)
namespace fastIO {
#define BUF_SIZE 100000
#define OUT_SIZE 100000
//fread->R
bool IOError = 0;
//inline char nc(){char ch=getchar();if(ch==-1)IOError=1;return ch;}
inline char nc() {
    static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
    if (p1 == pend) {
        p1 = buf;
        pend = buf + fread(buf, 1, BUF_SIZE, stdin);
        if (pend == p1) {
            IOError = 1;
            return -1;
        }
    }
    return *p1++;
}
inline bool blank(char ch) { return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t'; }
template <class T>
inline bool R(T &x) {
    bool sign = 0;
    char ch = nc();
    x = 0;
    for (; blank(ch); ch = nc())
        ;
    if (IOError)
        return false;
    if (ch == '-')
        sign = 1, ch = nc();
    for (; ch >= '0' && ch <= '9'; ch = nc())
        x = x * 10 + ch - '0';
    if (sign)
        x = -x;
    return true;
}
inline bool R(double &x) {
    bool sign = 0;
    char ch = nc();
    x = 0;
    for (; blank(ch); ch = nc())
        ;
    if (IOError)
        return false;
    if (ch == '-')
        sign = 1, ch = nc();
    for (; ch >= '0' && ch <= '9'; ch = nc())
        x = x * 10 + ch - '0';
    if (ch == '.') {
        double tmp = 1;
        ch = nc();
        for (; ch >= '0' && ch <= '9'; ch = nc())
            tmp /= 10.0, x += tmp * (ch - '0');
    }
    if (sign)
        x = -x;
    return true;
}
inline bool R(char *s) {
    char ch = nc();
    for (; blank(ch); ch = nc())
        ;
    if (IOError)
        return false;

```

```

    for (; !blank(ch) && !IOError; ch = nc())
        *s++ = ch;
    *s = 0;
    return true;
}
inline bool R(char &c) {
    c = nc();
    if (IOError) {
        c = -1;
        return false;
    }
    return true;
}
template <class T, class... U>
bool R(T &h, U &... t) { return R(h) && R(t...); }
#undef OUT_SIZE
#undef BUF_SIZE
}; // namespace fastIO
using namespace fastIO;
template <class T>
void _W(const T &x) { cout << x; }
void _W(const int &x) { printf("%d", x); }
void _W(const int64_t &x) { printf("%lld", x); }
void _W(const double &x) { printf("%.16f", x); }
void _W(const char &x) { putchar(x); }
void _W(const char *x) { printf("%s", x); }
template <class T, class U>
void _W(const pair<T, U> &x) { _W(x.F), putchar(' '), _W(x.S); }
template <class T>
void _W(const vector<T> &x) {
    for (auto i = x.begin(); i != x.end(); _W(*i++))
        if (i != x.cbegin()) putchar(' ');
}
void W() {}
template <class T, class... U>
void W(const T &head, const U &... tail) { _W(head), putchar(sizeof...(tail) ? ' ' : '\n'), W(tail...); }
#pragma endregion
//HDU - 5306 Gorgeous Sequence( jls)
const int maxn = 1e6 + 5;
int n, m, a[maxn];
struct segtree {
    int l, r, maxx, semax, cmax;
    ll sum;
} t[maxn << 2];
inline void pushup(int root) {
    tr.sum = lson.sum + rson.sum;
    tr.maxx = max(lson.maxx, rson.maxx);
    tr.semax = max(lson.semax, rson.semax);
    tr.cmax = 0;
    if (lson.maxx != rson.maxx) tr.semax = max(tr.semax, min(lson.maxx, rson.maxx));
    if (tr.maxx == lson.maxx) tr.cmax += lson.cmax;
    if (tr.maxx == rson.maxx) tr.cmax += rson.cmax;
}
void build(int root, int l, int r) {
    tr.l = l, tr.r = r;
    if (l == r) {
        tr.sum = tr.maxx = a[l];
        tr.cmax = 1;
        tr.semax = -1;
        return;
    }
    int mid = (l + r) >> 1;
    build(root << 1, l, mid);
    build(root << 1 | 1, mid + 1, r);
    pushup(root);
}

```

```

inline void dec_tag(int root, int x) { // maxxsum
    if (x >= tr.maxx) return;
    tr.sum += 1LL * (x - tr.maxx) * tr.cmax;
    tr.maxx = x;
}
inline void spread(int root) {
    dec_tag(root << 1, tr.maxx);
    dec_tag(root << 1 | 1, tr.maxx);
}
void update(int root, int l, int r, int x) {
    if (x >= tr.maxx) return; //
    if (l <= tr.l && tr.r <= r && x > tr.semax) { //
        dec_tag(root, x);
        return;
    }
    //
    spread(root);
    int mid = (tr.l + tr.r) >> 1;
    if (l <= mid) update(root << 1, l, r, x);
    if (r > mid) update(root << 1 | 1, l, r, x);
    pushup(root);
}
int qmax(int root, int l, int r) {
    if (l <= tr.l && tr.r <= r) return tr.maxx;
    spread(root);
    int mid = (tr.l + tr.r) >> 1;
    int maxx = 0;
    if (l <= mid) maxx = max(maxx, qmax(root << 1, l, r));
    if (r > mid) maxx = max(maxx, qmax(root << 1 | 1, l, r));
    return maxx;
}
ll qsum(int root, int l, int r) {
    if (l <= tr.l && tr.r <= r) return tr.sum;
    spread(root);
    ll ans = 0;
    int mid = (tr.l + tr.r) >> 1;
    if (l <= mid) ans += qsum(root << 1, l, r);
    if (r > mid) ans += qsum(root << 1 | 1, l, r);
    return ans;
}
int main() {
    int T;
    R(T);
    while (T--) {
        R(n, m);
        rep(i, 1, n) R(a[i]);
        build(1, 1, n);
        while (m--) {
            int op, l, r, x;
            R(op, l, r);
            if (op == 0) R(x), update(1, l, r, x); // a[i]=min(a[i],x)
            if (op == 1) W(qmax(1, l, r));
            if (op == 2) W(qsum(1, l, r));
        }
    }
}

```

1.18 segment_tree3.cpp

```

// #pragma GCC optimize(2)
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <vector>
using namespace std;

```

```

typedef long long ll;
const int maxn = 1e6 + 10;

ll n, m;
ll a[maxn];
struct segtree {
    int lc, rc; //
    int dat; //
} tr[maxn]; //
int root, tot; //

int build() //
{
    tot++;
    tr[tot].lc = tr[tot].rc = tr[tot].dat = 0; //
    return tot; // ()
}

void insert(int p, int l, int r, int val, int dat) // [l,r] val dat
{
    if (l == r) // l==r
    {
        tr[p].dat += dat; //
        return; //
    }
    int mid = (l + r) >> 1; //
    //
    if (val <= mid) // [l,mid]
    {
        if (!tr[p].lc)
            tr[p].lc = build(); //
        insert(tr[p].lc, l, mid, val, dat); //
    } else // [mid+1,r]
    {
        if (!tr[p].rc)
            tr[p].rc = build(); //
        insert(tr[p].rc, mid + 1, r, val, dat); //
    }
    tr[p].dat = tr[tr[p].lc].dat + tr[tr[p].rc].dat; //
}

ll query(int p, int l, int r, int ql, int qr) {
    if (ql <= l && qr >= r) // ,
    {
        return tr[p].dat; //
    }
    ll ans = 0; //
    int mid = (l + r) >> 1; //
    if (ql <= mid)
        ans += query(tr[p].lc, l, mid, ql, qr); //
    if (qr > mid)
        ans += query(tr[p].rc, mid + 1, r, ql, qr); //
    return ans; //
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    int T;
    cin >> T;
    for (int cas = 1; cas <= T; cas++) {
        cout << "Case " << cas << ":" << endl;

        root = 0, tot = 0;
        cin >> n;
        root = build();
        for (int i = 1; i <= n; i++)

```

```

        cin >> a[i], insert(root, 1, n, i, a[i]);
string s;
while (cin >> s) {
    if (s == "End")
        break;
    else if (s == "Query") {
        int l, r;
        cin >> l >> r;
        cout << query(root, 1, n, l, r) << endl;
    } else if (s == "Add") {
        int x, v;
        cin >> x >> v;
        insert(root, 1, n, x, v);
    } else if (s == "Sub") {
        int x, v;
        cin >> x >> v;
        insert(root, 1, n, x, -v);
    }
}
}
}
}

```

1.19 主席树.cpp

```

#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;
const int maxn = 1e5 + 5; //
int tot, n, m;
int sum[(maxn << 5) + 10], rt[maxn + 10], ls[(maxn << 5) + 10],
    rs[(maxn << 5) + 10];
int a[maxn + 10], ind[maxn + 10], len;
inline int getid(const int &val) { //
    return lower_bound(ind + 1, ind + len + 1, val) - ind;
}
int build(int l, int r) { //
    int root = ++tot;
    if (l == r)
        return root;
    int mid = (l + r) >> 1;
    ls[root] = build(l, mid);
    rs[root] = build(mid + 1, r);
    return root; //
}
int update(int k, int l, int r, int root) { //
    int dir = ++tot;
    ls[dir] = ls[root], rs[dir] = rs[root], sum[dir] = sum[root] + 1;
    if (l == r) return dir;
    int mid = (l + r) >> 1;
    if (k <= mid) ls[dir] = update(k, l, mid, ls[dir]);
    else rs[dir] = update(k, mid + 1, r, rs[dir]);
    return dir;
}
int query(int u, int v, int l, int r, int k) { //
    int mid = (l + r) >> 1, x = sum[ls[v]] - sum[ls[u]]; //
    if (l == r) return l;
    if (k <= x) //
        return query(ls[u], ls[v], l, mid, k);
    else //
        return query(rs[u], rs[v], mid + 1, r, k - x);
}
inline void init() {
    tot = 0;
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; ++i)
        scanf("%d", &a[i]);
}

```

```

memcpy(ind, a, sizeof ind);
sort(ind + 1, ind + n + 1);
len = unique(ind + 1, ind + n + 1) - ind - 1;
rt[0] = build(1, len);
for (int i = 1; i <= n; ++i)
    rt[i] = update(getid(a[i]), 1, len, rt[i - 1]);
}
int l, r, k;
inline int qmin(int k) { return ind[query(rt[l - 1], rt[r], 1, len, k)]; } // k
inline int qmax(int k) { return ind[query(rt[l - 1], rt[r], 1, len, r - l + 2 - k)]; } // k
} // k
inline void work() {
    while (m--) {
        scanf("%d%d%d", &l, &r, &k);
        printf("%d\n", ind[query(rt[l - 1], rt[r], 1, len, k)]); //
    }
}
int main() {
    init();
    work();
    return 0;
}

```

1.20 区间覆盖.cpp

```

#include <bits/stdc++.h>
#define rep(i, a, n) for (int i = a; i <= n; ++i)
#define per(i, a, n) for (int i = n; i >= a; --i)
#ifdef LOCAL
#include "Print.h"
#define de(...) W(' ', #__VA_ARGS__,) =", __VA_ARGS__"
#else
#define de(...)
#endif
using namespace std;
typedef long long ll;
const int maxn = 1e5 + 5;
int n, q, a[maxn];
vector<int> g[maxn];
int sz[maxn], id[maxn], idd[maxn], cnt;
void dfs(int u, int f) {
    sz[u] = 1, id[u] = ++cnt, idd[cnt] = u;
    for (auto v : g[u]) {
        if (v == f) continue;
        dfs(v, u);
        sz[u] += sz[v];
    }
}
struct segtree {
#define tr t[root]
#define lson t[root << 1]
#define rson t[root << 1 | 1]
    struct node {
        int l, r, maxx, minn;
        int add, cov;
    } t[maxn << 2];
    void build(int root, int l, int r) {
        tr.l = l, tr.r = r, tr.add = 0, tr.cov = -1;
        if (l == r) {
            tr.maxx = tr.minn = a[id[root]];
            return;
        }
        int mid = (l + r) >> 1;
        build(root << 1, l, mid);
        build(root << 1 | 1, mid + 1, r);
        pushup(root);
    }
}

```



```

void pushup(int root) {
    tr.maxx = max(lson.maxx, rson.maxx);
    tr.minn = min(lson.minn, rson.minn);
}
void spdCov(int root) {
    lson.minn = rson.minn = tr.cov;
    lson.maxx = rson.maxx = tr.cov;
    lson.cov = rson.cov = tr.cov;
}
void spdAdd(int root) {
    if (~lson.cov) {
        if (lson.l != lson.r) spdCov(root << 1);
        lson.cov = -1, lson.add = 0;
    }
    if (~rson.cov) {
        if (rson.l != rson.r) spdCov(root << 1 | 1);
        rson.cov = -1, rson.add = 0;
    }
    lson.minn += tr.add, rson.minn += tr.add;
    lson.maxx += tr.add, rson.maxx += tr.add;
    lson.add += tr.add, rson.add += tr.add;
}
void spread(int root) {
    if (~tr.cov) {
        if (tr.l != tr.r) spdCov(root);
        tr.cov = -1, tr.add = 0;
    }
    if (tr.add) {
        if (tr.l != tr.r) spdAdd(root);
        tr.add = 0;
    }
}
void cov(int root, int l, int r, int x) {
    spread(root);
    if (l <= tr.l && tr.r <= r) {
        tr.minn = x, tr.maxx = x;
        tr.add = 0, tr.cov = x;
        return;
    }
    int mid = (tr.l + tr.r) >> 1;
    if (l <= mid) cov(root << 1, l, r, x);
    if (r > mid) cov(root << 1 | 1, l, r, x);
    pushup(root);
}
void add(int root, int l, int r, int x) {
    spread(root);
    if (l <= tr.l && tr.r <= r) {
        tr.minn += x, tr.maxx += x;
        tr.add += x;
        return;
    }
    int mid = (tr.l + tr.r) >> 1;
    if (l <= mid) add(root << 1, l, r, x);
    if (r > mid) add(root << 1 | 1, l, r, x);
    pushup(root);
}
int qmax(int root, int l, int r) {
    spread(root);
    if (l <= tr.l && tr.r <= r) return tr.maxx;
    int mid = (tr.l + tr.r) >> 1, ans = 0;
    if (l <= mid) ans = max(ans, qmax(root << 1, l, r));
    if (r > mid) ans = max(ans, qmax(root << 1 | 1, l, r));
    return ans;
}
int qmin(int root, int l, int r) {
    spread(root);
    if (l <= tr.l && tr.r <= r) return tr.minn;
}

```

```

    int mid = (tr.l + tr.r) >> 1, ans = 2e9;
    if (l <= mid) ans = min(ans, qmin(root << 1, l, r));
    if (r > mid) ans = min(ans, qmin(root << 1 | 1, l, r));
    return ans;
}
} Tr;
inline void add(int u, int val) { Tr.add(1, id[u], id[u] + sz[u] - 1, val); }
inline void cov(int u, int val) { Tr.cov(1, id[u], id[u] + sz[u] - 1, val); }
inline int qry(int u) {
    int l = id[u], r = id[u] + sz[u] - 1;
    return Tr.qmax(1, l, r) - Tr.qmin(1, l, r);
}
int case_Test() {
    scanf("%d%d", &n, &q);
    rep(i, 1, n) scanf("%d", &a[i]);
    rep(i, 1, n - 1) {
        int u, v;
        scanf("%d%d", &u, &v);
        g[u].emplace_back(v);
        g[v].emplace_back(u);
    }
    dfs(1, 0), Tr.build(1, 1, n);
    while (q--) {
        int op, x, V;
        scanf("%d%d", &op, &x);
        if (op == 0) scanf("%d", &V), add(x, V);
        if (op == 1) scanf("%d", &V), cov(x, V);
        if (op == 2) printf("%d\n", qry(x));
    }
    return 0;
}
int main() {
#ifdef LOCAL
    freopen("/Users/chenjinglong/cpp_code/in.in", "r", stdin);
    freopen("/Users/chenjinglong/cpp_code/out.out", "w", stdout);
    clock_t start = clock();
#endif
    int _ = 1;
    // scanf("%d", &_);
    while (_--) case_Test();
#ifdef LOCAL
    printf("Time used: %.3lfs\n", (double)(clock() - start) / CLOCKS_PER_SEC);
#endif
    return 0;
}
// " " https://www.luogu.com.cn/problem/P4315

```

1.21 带权并查集.cpp

```

#include <bits/stdc++.h>
#define rep(i, a, n) for (int i = a; i <= n; ++i)
#define per(i, a, n) for (int i = n; i >= a; --i)
#ifdef LOCAL
#include "Print.h"
#define de(...) W(' ', __VA_ARGS__, " ") = __VA_ARGS__
#else
#define de(...)
#endif
using namespace std;
typedef long long ll;
const int maxn = 3e4 + 5;
int fa[maxn], sz[maxn], d[maxn]; // d
int findR(int x) {
    if (x == fa[x]) return x;
    int rt = findR(fa[x]);
    d[x] += d[fa[x]];
    return fa[x] = rt;
}

```

```

}
void link(int x, int y, int f) {
    int xx = findR(x), yy = findR(y);
    fa[xx] = yy, d[xx] += sz[yy];
    sz[yy] += sz[xx];
}
int query(int x, int y) {
    if (x == y) return 0;
    int xx = findR(x), yy = findR(y);
    if (xx != yy) return -1;
    return abs(d[x] - d[y]) - 1;
}
int main() {
    int T;
    scanf("%d", &T);
    rep(i, 1, maxn - 1) fa[i] = i, sz[i] = 1;
    while (T--) {
        char op[5]; int x, y;
        scanf("%s%d%d", op + 1, &x, &y);
        if (op[1] == 'M') link(x, y, 1);
        else printf("%d\n", query(x, y));
    }
    return 0;
}

```

1.22 替罪羊.cpp

```

#include <bits/stdc++.h>
#define rep(i, a, n) for (int i = a; i <= n; ++i)
#define per(i, a, n) for (int i = n; i >= a; --i)
#ifdef LOCAL
#include "Print.h"
#define de(...) W(['', #__VA_ARGS__, "], __VA_ARGS__)
#else
#define de(...)
#endif
using namespace std;
typedef long long ll;
const int maxn = 1e5 + 5;
namespace tzy {
#define tr t[root]
#define lson t[tr.lc]
#define rson t[tr.rc]
const double alpha = 0.75;
int cnt, Root;
struct node {
    int val, lc, rc;
    int num, sz, csz, dsz;
} t[maxn];
// root
void Calc(int root) {
    tr.sz = lson.sz + rson.sz + 1;
    tr.csz = lson.csz + rson.csz + tr.num;
    tr.dsz = lson.dsz + rson.dsz + (tr.num != 0);
}
// root
inline bool CanRbu(int root) {
    return tr.num && (max(lson.sz, rson.sz) >= alpha * tr.sz || tr.dsz <= alpha * tr.sz);
}
int ldr[maxn];
// root
void getLdr(int &len, int root) {
    if (!root) return;
    getLdr(len, tr.lc);
    if (tr.num) ldr[len++] = root;
    getLdr(len, tr.rc);
}

```

```

}
// ldr[] [l, r)
int lift(int l, int r) {
    int mid = (l + r) >> 1, R = ldr[mid];
    if (l >= r) return 0;
    t[R].lc = lift(l, mid);
    t[R].rc = lift(mid + 1, r);
    Calc(R);
    return R;
}
// root
void rebuild(int &root) {
    if (!CanRbu(root)) return;
    int len = 0;
    getLdr(len, root);
    root = lift(0, len);
}
// root val
void Insert(int &root, int val) {
    if (!root) {
        root = ++cnt;
        if (!Root) Root = 1;
        tr.val = val, tr.lc = tr.rc = 0;
        tr.num = tr.sz = tr.csz = tr.dsz = 1;
    } else {
        if (val == tr.val) tr.num++;
        else if (val < tr.val) Insert(tr.lc, val);
        else Insert(tr.rc, val);
        Calc(root), rebuild(root);
    }
}
// root val
void Del(int &root, int val) {
    if (!root) return;
    if (tr.val == val) {
        if (tr.num) tr.num--;
    } else {
        if (val < tr.val) Del(tr.lc, val);
        else Del(tr.rc, val);
    }
    Calc(root), rebuild(root);
}
// root val
int MyUprBd(int root, int val) {
    if (!root) return 1;
    if (val == tr.val && tr.num) return lson.csz + 1 + tr.num;
    if (val < tr.val) return MyUprBd(tr.lc, val);
    return lson.csz + tr.num + MyUprBd(tr.rc, val);
}
//
int MyUprGrT(int root, int val) {
    if (!root) return 0;
    if (val == tr.val) return lson.csz;
    if (val < tr.val) return MyUprGrT(tr.lc, val);
    return lson.csz + tr.num + MyUprGrT(tr.rc, val);
}
// root rnk
int Getnum(int root, int rnk) {
    if (!root) return 0;
    if (lson.csz < rnk && rnk <= lson.csz + tr.num) return tr.val;
    if (lson.csz >= rnk) return Getnum(tr.lc, rnk);
    return Getnum(tr.rc, rnk - lson.csz - tr.num);
}
inline void insert(int val) { Insert(Root, val); }
inline void del(int val) { Del(Root, val); }
inline int getnum(int rnk) { return Getnum(Root, rnk); }
inline int getrnk(int val) { return MyUprGrT(Root, val) + 1; }

```

```

inline int lowerRnk(int val) { return MyUprGrT(Root, val); }
inline int upperRnk(int val) { return MyUprBd(Root, val); }
inline int getpre(int val) { return getnum(lowerRnk(val)); }
inline int getnex(int val) { return getnum(upperRnk(val)); }
#undef tr
#undef lson
#undef rson
} // namespace tzy
int case_Test() {
    int _; scanf("%d", &_);
    while (_--) {
        int op, x;
        scanf("%d%d", &op, &x);
        if (op == 1) tzy::insert(x);
        if (op == 2) tzy::del(x);
        if (op == 3) printf("%d\n", tzy::getrnk(x));
        if (op == 4) printf("%d\n", tzy::getnum(x));
        if (op == 5) printf("%d\n", tzy::getpre(x));
        if (op == 6) printf("%d\n", tzy::getnex(x));
    }
    return 0;
}
int main() {
#ifdef LOCAL
    freopen("/Users/chenjinglong/Desktop/cpp_code/in.in", "r", stdin);
    freopen("/Users/chenjinglong/Desktop/cpp_code/out.out", "w", stdout);
    clock_t start = clock();
#endif
    int _ = 1;
    // scanf("%d", &_);
    while (_--) case_Test();
#ifdef LOCAL
    printf("Time used: %.3lfs\n", (double)(clock() - start) / CLOCKS_PER_SEC);
#endif
    return 0;
}

```

1.23 树剖.cpp

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

```
using i64 = long long;
```

```

struct Info {
    int c[2];
    i64 s[2];
    Info() : c{}, s{} {}
    Info(int x, int v) : Info() {
        c[x] = 1;
        s[x] = v;
    }
};

```

```

Info operator+(const Info &a, const Info &b) {
    Info c;
    c.c[0] = a.c[0] + b.c[0];
    c.c[1] = a.c[1] + b.c[1];
    c.s[0] = a.s[0] + b.s[0];
    c.s[1] = a.s[1] + b.s[1];
    return c;
}

```

```

void apply(Info &a, int b) {
    if (b) {
        std::swap(a.c[0], a.c[1]);
        std::swap(a.s[0], a.s[1]);
    }
}

```

```
}
```

```

void apply(int &a, int b) {
    a ^= b;
}

```

```

template<class Info, class Tag,
        class Merge = std::plus<Info>>
struct LazySegmentTree {
    const int n;
    const Merge merge;
    std::vector<Info> info;
    std::vector<Tag> tag;
    LazySegmentTree(int n) : n(n), merge(Merge()), info(4 << std::__lg(n)), tag(4 <<
        std::__lg(n)) {}
    LazySegmentTree(std::vector<Info> init) : LazySegmentTree(init.size()) {
        std::function<void(int, int, int)> build = [&](int p, int l, int r) {
            if (r - l == 1) {
                info[p] = init[l];
                return;
            }
            int m = (l + r) / 2;
            build(2 * p, l, m);
            build(2 * p + 1, m, r);
            pull(p);
        };
        build(1, 0, n);
    }
    void pull(int p) {
        info[p] = merge(info[2 * p], info[2 * p + 1]);
    }
    void apply(int p, const Tag &v) {
        ::apply(info[p], v);
        ::apply(tag[p], v);
    }
    void push(int p) {
        apply(2 * p, tag[p]);
        apply(2 * p + 1, tag[p]);
        tag[p] = Tag();
    }
    void modify(int p, int l, int r, int x, const Info &v) {
        if (r - l == 1) {
            info[p] = v;
            return;
        }
        int m = (l + r) / 2;
        push(p);
        if (x < m) {
            modify(2 * p, l, m, x, v);
        } else {
            modify(2 * p + 1, m, r, x, v);
        }
        pull(p);
    }
    void modify(int p, const Info &v) {
        modify(1, 0, n, p, v);
    }
    Info rangeQuery(int p, int l, int r, int x, int y) {
        if (l >= y || r <= x) {
            return Info();
        }
        if (l >= x && r <= y) {
            return info[p];
        }
        int m = (l + r) / 2;
        push(p);
        return merge(rangeQuery(2 * p, l, m, x, y), rangeQuery(2 * p + 1, m, r, x, y));
    }
}

```

```

    );
}
Info rangeQuery(int l, int r) {
    return rangeQuery(1, 0, n, l, r);
}
bool rangeApply(int p, int l, int r, int x, int y, const Tag &v) {
    if (l >= y || r <= x) {
        return true;
    }
    if (l >= x && r <= y && info[p].c[0] + info[p].c[1] == r - l) {
        apply(p, v);
        return true;
    }
    if (l >= x && r <= y && info[p].c[0] + info[p].c[1] == 0) {
        return false;
    }
    int m = (l + r) / 2;
    push(p);
    bool res;
    if (rangeApply(2 * p + 1, m, r, x, y, v)) {
        res = rangeApply(2 * p, l, m, x, y, v);
    } else {
        res = false;
    }
    pull(p);
    return res;
}
bool rangeApply(int l, int r, const Tag &v) {
    return rangeApply(1, 0, n, l, r, v);
}
};

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    int n;
    std::cin >> n;

    std::vector<std::vector<std::pair<int, int>>> adj(n);
    for (int i = 0; i < n - 1; i++) {
        int u, v;
        std::cin >> u >> v;
        u--;
        v--;

        adj[u].emplace_back(v, i + 1);
        adj[v].emplace_back(u, i + 1);
    }

    std::vector<int> id(n), parent(n, -1), dep(n), top(n), in(n), out(n), siz(n);
    int clk = 0;

    std::function<void(int)> dfs1 = [&](int u) {
        if (u > 0) {
            adj[u].erase(std::find(adj[u].begin(), adj[u].end(), std::pair(parent[u],
                id[u])));
        }
        siz[u] = 1;
        for (auto &e : adj[u]) {
            auto [v, i] = e;
            id[v] = i;
            parent[v] = u;
            dep[v] = dep[u] + 1;
            dfs1(v);
            siz[u] += siz[v];
            if (siz[v] > siz[adj[u][0].first]) {

```

```

                std::swap(adj[u][0], e);
            }
        }
    };
    dfs1(0);

    std::function<void(int)> dfs2 = [&](int u) {
        in[u] = clk++;
        for (auto [v, i] : adj[u]) {
            top[v] = v == adj[u][0].first ? top[u] : v;
            dfs2(v);
        }
        out[u] = clk;
    };
    dfs2(0);

    LazySegmentTree<Info, int> seg(n);
    seg.modify(0, Info(1, 0));

    while (true) {
        int op;
        std::cin >> op;

        if (op == 3) {
            break;
        }

        if (op == 1) {
            int x;
            std::cin >> x;
            x--;
            int s = 1;
            for (auto [v, i] : adj[x]) {
                s ^= seg.rangeQuery(in[v], in[v] + 1).c[1];
            }

            seg.modify(in[x], Info(s, id[x]));
            if (s == 1) {
                x = parent[x];
                while (x != -1) {
                    if (!seg.rangeApply(in[top[x]], in[x] + 1, 1)) {
                        break;
                    }
                    x = parent[top[x]];
                }
            }
        }

        auto info = seg.info[1];
        if (info.c[0] != info.c[1]) {
            std::cout << 0 << std::endl;
        } else if (op == 1) {
            std::cout << info.s[1] << std::endl;
        } else {
            std::vector<int> ans;
            for (int i = 0; i < n; i++) {
                if (seg.rangeQuery(in[i], in[i] + 1).c[1] == 1) {
                    ans.push_back(id[i]);
                }
            }
            std::sort(ans.begin(), ans.end());

            std::cout << ans.size();
            for (auto x : ans) {
                std::cout << " " << x;
            }
            std::cout << std::endl;

```

```

    }
}

return 0;
}

```

1.24 笛卡尔树.cpp

```

#include<bits/stdc++.h>
#define rep(i, a, n) for (int i = a; i <= n; ++i)
#define per(i, a, n) for (int i = n; i >= a; --i)
using namespace std;
typedef long long ll;
const int maxn = 1e7 + 5;
int n, a[maxn];
int ls[maxn], rs[maxn];
int top = 0;
// stack<int> st;
int st[maxn];
//
//
//
//
//
int main() {
    int n;
    scanf("%d", &n);
    rep(i, 1, n) {
        scanf("%d", &a[i]);
        // while (st.size() && a[st.top()] > a[i]) ls[i] = st.top(), st.pop();
        // if (st.size()) rs[st.top()] = i;
        // st.push(i);
        while (top && a[st[top]] > a[i]) ls[i] = st[top--];
        if (top) rs[st[top]] = i;
        st[++top] = i;
    }
    ll lans = 0, rans = 0;
    rep(i, 1, n) {
        lans ^= 1LL * i * (ls[i] + 1);
        rans ^= 1LL * i * (rs[i] + 1);
    }
    printf("%lld %lld\n", lans, rans);
    return 0;
}

```

1.25 轻重链剖分.cpp

```

// P3384
#pragma region
#include <algorithm>
#include <cmath>
#include <cstring>
#include <iomanip>
#include <iostream>
#include <map>
#include <queue>
#include <set>
#include <stack>
#include <string>
#include <vector>
using namespace std;
typedef long long ll;
#define tr t[root]
#define lson t[root << 1]
#define rson t[root << 1 | 1]
#define rep(i, a, n) for (int i = a; i <= n; ++i)
#define per(i, a, n) for (int i = n; i >= a; --i)
namespace fastIO {

```

```

#define BUF_SIZE 100000
#define OUT_SIZE 100000
//fread->R
bool IOError = 0;
//inline char nc(){char ch=getchar();if(ch==-1)IOError=1;return ch;}
inline char nc() {
    static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
    if (p1 == pend) {
        p1 = buf;
        pend = buf + fread(buf, 1, BUF_SIZE, stdin);
        if (pend == p1) {
            IOError = 1;
            return -1;
        }
    }
    return *p1++;
}

inline bool blank(char ch) { return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t'; }
template <class T>
inline bool R(T &x) {
    bool sign = 0;
    char ch = nc();
    x = 0;
    for (; blank(ch); ch = nc())
        ;
    if (IOError)
        return false;
    if (ch == '-')
        sign = 1, ch = nc();
    for (; ch >= '0' && ch <= '9'; ch = nc())
        x = x * 10 + ch - '0';
    if (sign)
        x = -x;
    return true;
}

inline bool R(double &x) {
    bool sign = 0;
    char ch = nc();
    x = 0;
    for (; blank(ch); ch = nc())
        ;
    if (IOError)
        return false;
    if (ch == '-')
        sign = 1, ch = nc();
    for (; ch >= '0' && ch <= '9'; ch = nc())
        x = x * 10 + ch - '0';
    if (ch == '.') {
        double tmp = 1;
        ch = nc();
        for (; ch >= '0' && ch <= '9'; ch = nc())
            tmp /= 10.0, x += tmp * (ch - '0');
    }
    if (sign)
        x = -x;
    return true;
}

inline bool R(char *s) {
    char ch = nc();
    for (; blank(ch); ch = nc())
        ;
    if (IOError)
        return false;
    for (; !blank(ch) && !IOError; ch = nc())
        *s++ = ch;
    *s = 0;
}

```

```

    return true;
}
inline bool R(char &c) {
    c = nc();
    if (IOerror) {
        c = -1;
        return false;
    }
    return true;
}
template <class T, class... U>
bool R(T &h, U &... t) { return R(h) && R(t...); }
#undef OUT_SIZE
#undef BUF_SIZE
}; // namespace fastIO
using namespace fastIO;
template <class T>
void _W(const T &x) { cout << x; }
void _W(const int &x) { printf("%d", x); }
void _W(const int64_t &x) { printf("%lld", x); }
void _W(const double &x) { printf("%.16f", x); }
void _W(const char &x) { putchar(x); }
void _W(const char *x) { printf("%s", x); }
template <class T, class U>
void _W(const pair<T, U> &x) { _W(x.F), putchar(' '), _W(x.S); }
template <class T>
void _W(const vector<T> &x) {
    for (auto i = x.begin(); i != x.end(); _W(*i++))
        if (i != x.cbegin()) putchar(' ');
}
void W() {}
template <class T, class... U>
void W(const T &head, const U &... tail) { _W(head), putchar(sizeof...(tail) ? ' ' : '\n'), W(tail...); }
#pragma endregion
const int maxn = 1e5 + 5;
int n, m, r, mod;
int w[maxn];
vector<int> g[maxn];
int fa[maxn], sz[maxn], dep[maxn], son[maxn];
int id[maxn], cnt, wt[maxn], top[maxn];
void init() {
    rep(i, 1, n) {
        g[i].clear();
        son[i] = 0;
    }
}
void dfs1(int u, int f, int deep) {
    dep[u] = deep, fa[u] = f, sz[u] = 1;
    for (auto v : g[u]) {
        if (v == f) continue;
        dfs1(v, u, deep + 1);
        sz[u] += sz[v];
        if (sz[v] > sz[son[u]]) son[u] = v;
    }
}
void dfs2(int u, int topf) {
    id[u] = ++cnt, wt[cnt] = w[u], top[u] = topf;
    if (!son[u]) return;
    dfs2(son[u], topf);
    for (auto v : g[u]) {
        if (v == fa[u] || v == son[u]) continue;
        dfs2(v, v);
    }
}
struct segtree {
    int l, r, val, lazy;

```

```

} t[maxn << 2];
void build(int root, int l, int r) {
    tr.l = l, tr.r = r, tr.lazy = 0;
    if (l == r) {
        tr.val = wt[l] % mod;
        return;
    }
    int mid = (l + r) >> 1;
    build(root << 1, l, mid);
    build(root << 1 | 1, mid + 1, r);
    tr.val = (lson.val + rson.val) % mod;
}
void spread(int root) {
    if (tr.lazy) {
        lson.val = (lson.val + tr.lazy * (lson.r - lson.l + 1)) % mod;
        rson.val = (rson.val + tr.lazy * (rson.r - rson.l + 1)) % mod;
        lson.lazy = (lson.lazy + tr.lazy) % mod;
        rson.lazy = (rson.lazy + tr.lazy) % mod;
        tr.lazy = 0;
    }
}
int query(int root, int l, int r) {
    if (l <= tr.l && tr.r <= r) return tr.val % mod;
    spread(root);
    int ans = 0;
    int mid = (tr.l + tr.r) >> 1;
    if (l <= mid) ans = (ans + query(root << 1, l, r)) % mod;
    if (r > mid) ans = (ans + query(root << 1 | 1, l, r)) % mod;
    return ans;
}
void update(int root, int l, int r, int x) {
    if (l <= tr.l && tr.r <= r) {
        tr.val = (tr.val + x * (tr.r - tr.l + 1)) % mod;
        tr.lazy = (tr.lazy + x) % mod;
        return;
    }
    spread(root);
    int mid = (tr.l + tr.r) >> 1;
    if (l <= mid) update(root << 1, l, r, x);
    if (r > mid) update(root << 1 | 1, l, r, x);
    tr.val = (lson.val + rson.val) % mod;
}
int qSon(int x) { return query(1, id[x], id[x] + sz[x] - 1); }
void updSon(int x, int k) { update(1, id[x], id[x] + sz[x] - 1, k); }
int qRange(int x, int y) {
    int ans = 0;
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        ans = (ans + query(1, id[top[x]], id[x])) % mod;
        x = fa[top[x]];
    }
    if (dep[x] > dep[y]) swap(x, y);
    ans = (ans + query(1, id[x], id[y])) % mod;
    return ans;
}
void updRange(int x, int y, int k) {
    k %= mod;
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        update(1, id[top[x]], id[x], k);
        x = fa[top[x]];
    }
    if (dep[x] > dep[y]) swap(x, y);
    update(1, id[x], id[y], k);
}
int main() {
    R(n, m, r, mod);

```

```

rep(i, 1, n) R(w[i]);
rep(i, 1, n - 1) {
    int u, v;
    R(u, v);
    g[u].push_back(v);
    g[v].push_back(u);
}
dfs1(r, 0, 1);
dfs2(r, r);
build(1, 1, n);
while (m--) {
    int op, x, y, z;
    R(op);
    if (op == 1)
        R(x, y, z), updRange(x, y, z);
    else if (op == 2)
        R(x, y), W(qRange(x, y));
    else if (op == 3)
        R(x, y), updSon(x, y);
    else
        R(x), W(qSon(x));
}
}

```

2 Geometry

2.1 Circle.cpp

```
#include "PolygonAndConvex.cpp"
```

```

double sqr(double x) { return x * x; }
double mysqrt(double n) {
    return sqrt(max(0.0, n));
} // sqrt(-eps)

struct Circle {
    Point o;
    double r;
    Circle(Point o = Point(), double r = 0) : o(o), r(r) {}
    bool operator==(const Circle &c) { return o == c.o && !sgn(r - c.r); }
    double area() { return PI * r * r; }
    double perimeter() { return r * PI * 2; }
    //
    bool pointIn(const Point &p) { return sgn((p - o).norm() - r) < 0; }
    //
    friend int isLineCircleIntersection(Line L, Circle c) {
        return L.disPointLine(c.o) < c.r + eps;
    }
    //
    friend int isSegCircleIntersection(Line L, Circle c) {
        double t1 = dis(c.o, L.s) - c.r, t2 = dis(c.o, L.t) - c.r;
        Point t = c.o;
        if (t1 < eps || t2 < eps) return t1 > -eps || t2 > -eps;
        t.x += L.s.y - L.t.y;
        t.y += L.t.x - L.s.x;
        return det(L.s - t, c.o - t) * det(L.t - t, c.o - t) < eps && L.disPointLine(
            c.o) < c.r + eps;
    }
    //
    friend int isCirCirIntersection(Circle c1, Circle c2) {
        return dis(c1.o, c2.o) < c1.r + c2.r + eps &&
            dis(c1.o, c2.o) > fabs(c1.r - c2.r) - eps;
    }
    //
    friend int isCirCirContain(Circle c1, Circle c2) {
        return sgn(dis(c1.o, c2.o) + min(c1.r, c2.r) - max(c1.r, c2.r)) <= 0;
    }
}

```

```

//      p , p , p
friend Point dotPointCircle(Point p, Circle C) {
    Point u, v, c = C.o;
    if (dis(p, c) < eps) return p;
    u.x = c.x + C.r * fabs(c.x - p.x) / dis(c, p);
    u.y = c.y + C.r * fabs(c.y - p.y) / dis(c, p) * ((c.x - p.x) * (c.y - p.y) <
        0 ? -1 : 1);
    v.x = c.x - C.r * fabs(c.x - p.x) / dis(c, p);
    v.y = c.y - C.r * fabs(c.y - p.y) / dis(c, p) * ((c.x - p.x) * (c.y - p.y) <
        0 ? -1 : 1);
    return dis(u, p) < dis(v, p) ? u : v;
}
//      P=A+t*(B-A)      t
friend vector<Point> segCircleIntersection(const Line &l, const Circle &c) {
    double dx = l.t.x - l.s.x, dy = l.t.y - l.s.y;
    double A = dx * dx + dy * dy;
    double B = 2 * dx * (l.s.x - c.o.x) + 2 * dy * (l.s.y - c.o.y);
    double C = sqr(l.s.x - c.o.x) + sqr(l.s.y - c.o.y) - sqr(c.r);
    double delta = B * B - 4 * A * C;
    vector<Point> res;
    if (A < eps) return res;
    if (sgn(delta) >= 0) { // or delta > -eps ?
        //      delta-      epsmysqrt
        double w1 = (-B - mysqrt(delta)) / (2 * A);
        double w2 = (-B + mysqrt(delta)) / (2 * A);
        if (sgn(w1 - 1) <= 0 && sgn(w1) >= 0) {
            res.push_back(l.s + w1 * (l.t - l.s));
        }
        if (sgn(w2 - 1) <= 0 && sgn(w2) >= 0 && fabs(w1 - w2) > eps) {
            res.push_back(l.s + w2 * (l.t - l.s));
        }
    }
    return res;
}
//
friend vector<Point> lineCircleIntersection(const Line &l, const Circle &c) {
    double dx = l.t.x - l.s.x, dy = l.t.y - l.s.y;
    double A = dx * dx + dy * dy;
    double B = 2 * dx * (l.s.x - c.o.x) + 2 * dy * (l.s.y - c.o.y);
    double C = sqr(l.s.x - c.o.x) + sqr(l.s.y - c.o.y) - sqr(c.r);
    double delta = B * B - 4 * A * C;
    vector<Point> res;
    if (A < eps) return res;
    if (sgn(delta) >= 0) { // or delta > -eps ?
        double w1 = (-B - mysqrt(delta)) / (2 * A);
        double w2 = (-B + mysqrt(delta)) / (2 * A);
        res.push_back(l.s + w1 * (l.t - l.s));
        if (fabs(w1 - w2) > eps) res.push_back(l.s + w2 * (l.t - l.s));
    }
    return res;
}
//
friend vector<Point> cirCirIntersection(Circle a, Circle b) {
    Point c1 = a.o;
    vector<Point> vec;
    if (dis(a.o, b.o) + eps > a.r + b.r &&
        dis(a.o, b.o) < fabs(a.r - b.r) + eps)
        return vec;
    //
    Line L;
    double t = (1.0 + (sqr(a.r) - sqr(b.r)) / sqr(dis(a.o, b.o))) / 2;
    L.s = c1 + (b.o - a.o) * t;
    L.t.x = L.s.x + a.o.y - b.o.y;
    L.t.y = L.s.y - a.o.x + b.o.x;
    return lineCircleIntersection(L, a);
}
//      pangle
//      (o,r)

```

```

friend vector<Point> tangentPointCircle(Point poi, Circle C) {
    Point o = C.o;
    double r = C.r;
    vector<Point> vec;
    double dist = (poi - o).norm();
    if (dist < r - eps) return vec;
    if (fabs(dist - r) < eps) {
        vec.push_back(poi);
        return vec;
    }
    Point res1, res2;
    double line =
        sqrt((poi.x - o.x) * (poi.x - o.x) + (poi.y - o.y) * (poi.y - o.y));
    double angle = acos(r / line);
    Point unitVector, lin;
    lin.x = poi.x - o.x;
    lin.y = poi.y - o.y;
    unitVector.x = lin.x / sqrt(lin.x * lin.x + lin.y * lin.y) * r;
    unitVector.y = lin.y / sqrt(lin.x * lin.x + lin.y * lin.y) * r;
    res1 = rotate(unitVector, -angle) + o;
    res2 = rotate(unitVector, angle) + o;
    vec.push_back(res1);
    vec.push_back(res2);
    return vec;
}

// a->b
double sectorArea(const Point &a, const Point &b) const {
    double theta = atan2(a.y, a.x) - atan2(b.y, b.x);
    while (theta < 0) theta += 2 * PI;
    while (theta > 2.0 * PI) theta -= 2 * PI;
    theta = min(theta, 2.0 * PI - theta);
    return sgn(det(a, b)) * theta * r * r / 2.0;
}

// AB a->b
double areaSegCircle(const Line &L) const {
    Point a = L.s, b = L.t;
    vector<Point> p = segCircleIntersection(Line(a, b), *this);
    bool ina = sgn((a - o).norm() - r) < 0;
    bool inb = sgn((b - o).norm() - r) < 0;
    if (ina) {
        if (inb)
            return det(a - o, b - o) / 2;
        else
            return det(a - o, p[0] - o) / 2 + sectorArea(p[0] - o, b - o);
    } else {
        if (inb)
            return det(p[0] - o, b - o) / 2 + sectorArea(a - o, p[0] - o);
        else {
            if (p.size() == 2)
                return sectorArea(a - o, p[0] - o) +
                    sectorArea(p[1] - o, b - o) +
                    det(p[0] - o, p[1] - o) / 2;
            else
                return sectorArea(a - o, b - o);
        }
    }
}

// +eps
friend double areaPolygonCircle(const Circle &c, const Polygon &a) {
    int n = a.p.size();

    double ans = 0;
    for (int i = 0; i < n; ++i) {
        if (sgn(det(a.p[i] - c.o, a.p[_next(i)] - c.o)) == 0) {
            continue;
        }

```

```

        ans += c.areaSegCircle((a.p[i], a.p[_next(i)]));
    }
    return ans;
}

//
friend double areaCircleCircle(const Circle &A, const Circle &B) {
    double ans = 0.0;
    Circle M = (A.r > B.r) ? A : B;
    Circle N = (A.r > B.r) ? B : A;
    double D = dis(M.o, N.o);
    if ((D < M.r + N.r) && (D > M.r - N.r)) {
        double alpha = 2.0 * acos((M.r * M.r + D * D - N.r * N.r) / (2.0 * M.r * D));
        double beta = 2.0 * acos((N.r * N.r + D * D - M.r * M.r) / (2.0 * N.r * D));
        ans = (alpha / (2 * PI)) * M.area() + (beta / (2 * PI)) * N.area() -
            0.5 * M.r * M.r * sin(alpha) - 0.5 * N.r * N.r * sin(beta);
    } else if (D <= M.r - N.r) {
        ans = N.area();
    }
    return ans;
}

//
Circle getCircle3(const Point &p0, const Point &p1, const Point &p2) {
    double a1 = p1.x - p0.x, b1 = p1.y - p0.y, c1 = (a1 * a1 + b1 * b1) / 2;
    double a2 = p2.x - p0.x, b2 = p2.y - p0.y, c2 = (a2 * a2 + b2 * b2) / 2;
    double d = a1 * b2 - a2 * b1;
    Point o(p0.x + (c1 * b2 - c2 * b1) / d, p0.y + (a1 * c2 - a2 * c1) / d);
    return Circle(o, (o - p0).norm());
}

//
Circle getCircle2(const Point &p0, const Point &p1) {
    Point o((p0.x + p1.x) / 2, (p0.y + p1.y) / 2);
    return Circle(o, (o - p0).norm());
}

// random_shuffle
Circle minCirCover(vector<Point> &a) {
    int n = a.size();
    Circle c(a[0], 0);
    for (int i = 1; i < n; ++i) {
        if (!c.pointIn(a[i])) {
            c.o = a[i];
            c.r = 0;
            for (int j = 0; j < i; ++j) {
                if (!c.pointIn(a[j])) {
                    c = getCircle2(a[i], a[j]);
                    for (int k = 0; k < j; ++k) {
                        if (!c.pointIn(a[k])) {
                            c = getCircle3(a[i], a[j], a[k]);
                        }
                    }
                }
            }
        }
    }
    return c;
}

//
friend double lengthSegInCircle(Line a, Circle c) {
    if (c.pointIn(a.s) && c.pointIn(a.t)) return a.norm();
    vector<Point> vec = segCircleIntersection(a, c);
    if (vec.size() == 0) return 0;
    if (vec.size() == 1) {
        if (c.pointIn(a.s)) return dis(vec[0], a.s);
        if (c.pointIn(a.t)) return dis(vec[0], a.t);
        return 0;
    }
}

```



```

    }
    return dis(vec[0], vec[1]);
}
//
friend double lengthPolygonInCircle(Polygon a, Circle c) {
    double ans = 0;
    for (int i = 0; i < a.n; ++i) {
        Line li;
        li.s = a.p[i];
        li.t = a.p[(i + 1) % a.n];
        ans += lengthSegInCircle(li, c);
    }
    return ans;
}
//      ba
friend double lengthCircleInCircle(Circle a, Circle b) {
    if (a.r > b.r && a.r - b.r + eps > dis(a.o, b.o)) return b.perimeter();
    vector<Point> vec = cirCirIntersection(a, b);
    if (vec.size() < 2) return 0;
    // Line l1 = (vec[0], b.o), l2 = (vec[1], b.o);
    double ans = b.r * arg_3(vec[0], b.o, vec[1]);
    if (b.r >= a.r || !a.pointIn(b.o)) return b.r * ans;
    return b.perimeter() - ans;
}
};

```

2.2 HalfPlane.cpp

```
#include "PolygonAndConvex.cpp"
```

```
const int inf = 1e9;
```

```

struct HalfPlane : public Line { //
    // ax + by + c <= 0
    double a, b, c;
    // s->t
    HalfPlane(const Point &s = Point(), const Point &t = Point()) : Line(s, t) {
        a = t.y - s.y;
        b = s.x - t.x;
        c = det(t, s);
    }
    HalfPlane(double a, double b, double c) : a(a), b(b), c(c) {}
    //      p
    double calc(const Point &p) const { return p.x * a + p.y * b + c; }
    //      lineIntersection4abc
    friend Point halfxLine(const HalfPlane &h, const Line &l) {
        Point res;
        double t1 = h.calc(l.s), t2 = h.calc(l.t);
        res.x = (t2 * l.s.x - t1 * l.t.x) / (t2 - t1);
        res.y = (t2 * l.s.y - t1 * l.t.y) / (t2 - t1);
        return res;
    }
    //      abc
    friend Point halfxHalf(const HalfPlane &h1, const HalfPlane &h2) {
        return Point(
            (h1.b * h2.c - h1.c * h2.b) / (h1.a * h2.b - h2.a * h1.b) + eps,
            (h1.a * h2.c - h2.a * h1.c) / (h1.b * h2.a - h1.a * h2.b) + eps);
    }
    //      (cut)
    friend Convex halfxConvex(const HalfPlane &h, const Convex &c) {
        Convex res;
        for (int i = 0; i < c.n; ++i) {
            if (h.calc(c.p[i]) < -eps)
                res.p.push_back(c.p[i]);
            else {
                int j = i - 1;
                if (j < 0) j = c.n - 1;

```

```

                if (h.calc(c.p[j]) < -eps)
                    res.p.push_back(halfxLine(h, Line(c.p[j], c.p[i])));
                j = i + 1;
                if (j == c.n) j = 0;
                if (h.calc(c.p[j]) < -eps) {
                    res.p.push_back(halfxLine(h, Line(c.p[i], c.p[j])));
                }
            }
        }
        res.n = res.p.size();
        return res;
    }
    //
    friend int satisfy(const Point &p, const HalfPlane &h) {
        return sgn(det(p - h.s, h.t - h.s)) <= 0;
    }
    friend bool operator<(const HalfPlane &h1, const HalfPlane &h2) {
        int res = sgn(h1.vec().arg() - h2.vec().arg());
        return res == 0 ? satisfy(h1.s, h2) : res < 0;
    }
    //
    friend Convex halfx(vector<HalfPlane> &v) {
        sort(v.begin(), v.end());
        deque<HalfPlane> q;
        deque<Point> ans;
        q.push_back(v[0]);
        for (int i = 1; i < v.size(); ++i) {
            if (sgn(v[i].vec().arg() - v[i - 1].vec().arg()) == 0) continue;
            while (ans.size() > 0 && !satisfy(ans.back(), v[i])) {
                ans.pop_back();
                q.pop_back();
            }
            while (ans.size() > 0 && !satisfy(ans.front(), v[i])) {
                ans.pop_front();
                q.pop_front();
            }
            ans.push_back(lineIntersection(q.back(), v[i]));
            q.push_back(v[i]);
        }
        while (ans.size() > 0 && !satisfy(ans.back(), q.front())) {
            ans.pop_back();
            q.pop_back();
        }
        while (ans.size() > 0 && !satisfy(ans.front(), q.back())) {
            ans.pop_front();
            q.pop_front();
        }
        ans.push_back(lineIntersection(q.back(), q.front()));
        Convex c(ans.size());
        int i = 0;
        for (deque<Point>::iterator it = ans.begin(); it != ans.end(); ++it, ++i) {
            c.p[i] = *it;
        }
        return c;
    }
};
//
Convex core(const Polygon &a) {
    Convex res;
    res.p.push_back(Point(-inf, -inf));
    res.p.push_back(Point(inf, -inf));
    res.p.push_back(Point(inf, inf));
    res.p.push_back(Point(-inf, inf));
    res.n = 4;
    for (int i = 0; i < a.n; i++) {
        res = halfxConvex(HalfPlane(a.p[i], a.p[(i + 1) % a.n]), res);
    }
}

```

```

    }
    return res;
}
//
Convex convexxConvex(Convex &c1, Convex &c2) {
    vector<HalfPlane> h;
    for (int i = 0; i < c1.p.size(); ++i)
        h.push_back(HalfPlane(c1.p[i], c1.p[(i + 1) % c1.p.size()]));
    for (int i = 0; i < c2.p.size(); i++)
        h.push_back(HalfPlane(c2.p[i], c2.p[(i + 1) % c2.p.size()]));
    return halfx(h);
}

2.3 Line.cpp

#include "Point.cpp"

const double PI = acos(-1);
struct Line {
    int id;
    Point s, t;
    Line(const Point &s = Point(), const Point &t = Point()) : s(s), t(t) {}

    Point vec() const { return t - s; } //
    double norm() const { return vec().norm(); } //
    //
    bool pointOnLine(const Point &p) {
        return sgn(det(p - s, t - s)) == 0;
    }
    //
    bool pointOnSeg(const Point &p) {
        return pointOnLine(p) && sgn(dot(p - s, p - t)) <= 0;
    }
    //
    bool pointOnSegInterval(const Point &p) {
        return pointOnLine(p) && sgn(dot(p - s, p - t) < 0);
    }
    //
    Point pedalPointLine(const Point &p) {
        return s + vec() * ((dot(p - s, vec()) / norm()) / norm());
    }
    //
    double disPointLine(const Point &p) {
        return fabs(det(p - s, vec()) / norm());
    }
    //
    double disPointSeg(const Point &p) {
        if (sgn(dot(p - s, t - s)) < 0) return (p - s).norm();
        if (sgn(dot(p - t, s - t)) < 0) return (p - t).norm();
        return disPointLine(p);
    }
    //
    // p ONLINELEFTRIGHT 0 -1 1
    int relation(const Point &p) { return sgn(det(t - s, p - s)); }
    // a, b
    bool sameSide(const Point &a, const Point &b) {
        return relation(a) == relation(b);
    }
    //
    // p
    Point symPoint(const Point &p) {
        return 2.0 * s - p + 2.0 * (t - s) * dot(p - s, t - s) / ((t.x - s.x) * (t.x
        - s.x) + (t.y - s.y) * (t.y - s.y));
    }
    //
    friend bool isParallel(const Line &l1, const Line &l2) {
        return sgn(det(l1.vec(), l2.vec())) == 0;
    }
    //

```

```

    friend Point lineIntersection(const Line &l1, const Line &l2) {
        double s1 = det(l1.s - l2.s, l2.vec());
        double s2 = det(l1.t - l2.s, l2.vec());
        return (l1.t * s1 - l1.s * s2) / (s1 - s2);
    }
    //
    friend Point getLineIntersection(const Line &u, const Line &v) {
        return u.s + (u.t - u.s) * det(u.s - v.s, v.s - v.t) /
            det(u.s - u.t, v.s - v.t);
    }
    //
    // l1l2
    friend bool isLineSegIntersection(Line l1, Line l2) {
        return l1.relation(l2.s) * l1.relation(l2.t) <= 0;
    }
    //
    friend bool isSegIntersection(Line l1, Line l2) {
        if (!sgn(det(l2.s - l1.s, l1.vec())) &&
            !sgn(det(l2.t - l1.t, l1.vec()))) {
            return l1.pointOnSeg(l2.s) || l1.pointOnSeg(l2.t) ||
                l2.pointOnSeg(l1.s) || l2.pointOnSeg(l1.t);
        }
        return !l1.sameSide(l2.s, l2.t) && !l2.sameSide(l1.s, l1.t);
    }
    //
    // ,
    // , , 1, 2, 0
    friend int segSegIntersection(Line l1, Line l2, Point &p) {
        Point a, b, c, d;
        a = l1.s;
        b = l1.t;
        c = l2.s;
        d = l2.t;
        double s1, s2, s3, s4;
        int d1, d2, d3, d4;
        d1 = sgn(s1 = det(b - a, c - a)); // l1.relation(l2.s);
        d2 = sgn(s2 = det(b - a, d - a)); // l1.relation(l2.t);
        d3 = sgn(s3 = det(d - c, a - c)); // l2.relation(l1.s);
        d4 = sgn(s4 = det(d - c, b - c)); // l2.relation(l1.t);

        //
        if (d1 * d2 < 0 && d3 * d4 < 0) {
            p.x = (c.x * s2 - d.x * s1) / (s2 - s1);
            p.y = (c.y * s2 - d.y * s1) / (s2 - s1);
            return 1;
        }
        //
        // d1 == 0, a, b, c;
        // sgn(dot(a - c, b - c)) < 0, cab
        // sgn(dot(a - c, b - c)) == 0, cabab
        // sgn(dot(a - c, b - c)) > 0, cab
        if ((d1 == 0 && sgn(dot(a - c, b - c)) <= 0) ||
            (d2 == 0 && sgn(dot(a - d, b - d)) <= 0) ||
            (d3 == 0 && sgn(dot(c - a, d - a)) <= 0) ||
            (d4 == 0 && sgn(dot(c - b, d - b)) <= 0)) {
            return 2;
        }
        return 0;
    }
    //
    // (-d) d
    friend Line move(const Line &l, const double &d) {
        Point t = l.vec();
        t = t / t.norm();
        t = rotate(t, PI / 2);
        return Line(l.s + t * d, l.t + t * d);
    }
}

```

```
// l1 l2
friend double disSegSeg(Line &l1, Line &l2) {
    double d1, d2, d3, d4;
    if (isSegIntersection(l1, l2))
        return 0;
    else {
        d1 = l2.disPointSeg(l1.s);
        d2 = l2.disPointSeg(l1.t);
        d3 = l1.disPointSeg(l2.s);
        d4 = l1.disPointSeg(l2.t);
        return min(min(d1, d2), min(d3, d4));
    }
}
// [0, PI]
friend double argLineLine(Line l1, Line l2) {
    Point u = l1.vec();
    Point v = l2.vec();
    return acos(dot(u, v) / (u.norm() * v.norm()));
}
};
```

2.4 Point.cpp

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

using namespace std;
using ll = long long;

const double eps = 1e-8;

int sgn(double x) { return abs(x) < eps ? 0 : (x > 0 ? 1 : -1); }

struct Point { // Point & Vector
    double x, y;
    Point(const double &x = 0, const double &y = 0) : x(x), y(y) {}

    friend Point operator+(const Point &a, const Point &b) {
        return Point(a.x + b.x, a.y + b.y);
    }
    friend Point operator-(const Point &a, const Point &b) {
        return Point(a.x - b.x, a.y - b.y);
    }
    friend Point operator*(const double &c, const Point &a) {
        return Point(c * a.x, c * a.y);
    }
    friend Point operator*(const Point &a, const double &c) {
        return Point(c * a.x, c * a.y);
    }
    friend Point operator/(const Point &a, const double &c) {
        return Point(a.x / c, a.y / c);
    }
    friend Point rotate(const Point &v, double theta) { // theta
        return Point(v.x * cos(theta) - v.y * sin(theta),
                     v.x * sin(theta) + v.y * cos(theta));
    }
    friend Point rotateAroundPoint(Point &v, Point &p, double theta) {
        return rotate(v - p, theta) + p;
    }
    friend bool operator==(const Point &a, const Point &b) {
        return !sgn(a.x - b.x) && !sgn(a.y - b.y);
    }
    friend bool operator<(const Point &a, const Point &b) {
        return sgn(a.x - b.x) < 0 || (!sgn(a.x - b.x) && sgn(a.y - b.y) < 0);
    }
    //
    double norm() { return sqrt(x * x + y * y); }
    //
```

```
friend double det(const Point &a, const Point &b) {
    return a.x * b.y - a.y * b.x;
}
//
friend double dot(const Point &a, const Point &b) {
    return a.x * b.x + a.y * b.y;
}
//
friend double dis(const Point &a, const Point &b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}
friend Point intersection(Point u1, Point u2, Point v1, Point v2) { // ,
    return u1 + (u2 - u1) * det(u1 - v1, v1 - v2) / det(u1 - u2, v1 - v2);
}
double arg() { return atan2(y, x); } //
friend double arg_2(Point u, Point v) {
    return acos(dot(u, v) / (u.norm() * v.norm()));
} //
friend double arg_3(const Point &a, const Point &b, const Point &c) {
    return arg_2(a - b, c - b);
} // abc
};
```

2.5 PolygonAndConvex.cpp

```
#include "Line.cpp"

struct Polygon {
#define _next(i) ((i + 1) % n)
    int n;
    vector<Point> p;

    Polygon(vector<Point> &v) : p(v) { n = p.size(); }
    Polygon(int n = 0) : n(n) { p.resize(n); }

    void addPoint(Point &a) {
        p.push_back(a);
        n++;
    }
    //
    double perimeter() {
        double sum = 0;
        for (int i = 0; i < n; ++i) sum += (p[_next(i)] - p[i]).norm();
        return sum;
    }
    //
    double area() {
        double sum = 0;
        for (int i = 0; i < n; ++i) sum += det(p[i], p[_next(i)]);
        return fabs(sum) / 2;
    }
    // eps
    // 0, 1, 2
    int pointIn(const Point &t) {
        int num = 0;
        for (int i = 0; i < n; ++i) {
            if (Line(p[i], p[_next(i)]).pointOnSeg(t)) return 2;
            int k = sgn(det(p[_next(i)] - p[i], t - p[i]));
            int d1 = sgn(p[i].y - t.y);
            int d2 = sgn(p[_next(i)].y - t.y);
            if (k > 0 && d1 <= 0 && d2 > 0) num++;
            if (k < 0 && d2 <= 0 && d1 > 0) num--;
        }
        return num % 2;
    }
    //
    Point baryCenter() {
        Point ans;
```

```

    if (sgn(area()) == 0) return ans;
    for (int i = 0; i < n; ++i)
        ans = ans + (p[i] + p[_next(i)]) * det(p[i], p[_next(i)]);
    return ans / area() / 6 + eps; // eps
}
//
bool isConvex() { // 3
    int s[3] = {1, 1, 1};
    for (int i = 0; i < n && (s[0] || s[2]) && s[1]; ++i) {
        s[1 + sgn(det(p[_next(i)] - p[i], p[_next(_next(i)] - p[i])))] = 0;
    }
    return (s[0] || s[2]) && s[1];
}
bool isConvex_3() { // 3
    int s[3] = {1, 1, 1};
    for (int i = 0; i < n && (s[0] || s[2]); ++i) {
        s[1 + sgn(det(p[_next(i)] - p[i], p[_next(_next(i)] - p[i])))] = 0;
    }
    return (s[0] || s[2]);
}
//
long long borderPointNum() {
    long long num = 0;
    for (int i = 0; i < n; ++i) {
        num += gcd((long long)fabs(p[_next(i)].x - p[i].x),
                    (long long)fabs(p[_next(i)].y - p[i].y));
    }
    return num;
}
//
long long inSidePointNum() {
    return (long long)(area()) + 1 - borderPointNum() / 2;
}
// p l1l2
inline int dotOnlineIn(Point p, Point l1, Point l2) {
    return sgn(det(p - l2, l1 - l2)) && (l1.x - p.x) * (l2.x - p.x) < eps &&
        (l1.y - p.y) * (l2.y - p.y) < eps;
}
// , , 1
int insidePolygon(Line l) {
    vector<Point> t;
    Point tt, l1 = l.s, l2 = l.t;
    if (!pointIn(l.s) || !pointIn(l.t)) return 0;
    for (int i = 0; i < n; ++i) {
        if (l.sameSide(p[i], p[(i + 1) % n]) &&
            l.sameSide(p[i], p[(i + 1) % n]))
            return 0;
        else if (dotOnlineIn(l1, p[i], p[(i + 1) % n]))
            t.push_back(l1);
        else if (dotOnlineIn(l2, p[i], p[(i + 1) % n]))
            t.push_back(l2);
        else if (dotOnlineIn(p[i], l1, l2))
            t.push_back(p[i]);
    }
    for (int i = 0; i < t.size(); ++i) {
        for (int j = i + 1; j < t.size(); ++j) {
            if (!pointIn((t[i] + t[j]) / 2)) return 0;
        }
    }
    return 1;
}
};

struct Convex : public Polygon {
    Convex(int n = 0) : Polygon(n) {}
    Convex(vector<Point> &a) { // n
        Convex res(a.size() * 2 + 7);

```

```

        sort(a.begin(), a.end());
        a.erase(unique(a.begin(), a.end()), a.end()); //
        int m = 0;
        for (int i = 0; i < a.size(); ++i) {
            // <0 3<=0
            while (m > 1 && sgn(det(res.p[m - 1] - res.p[m - 2], a[i] - res.p[m - 2])
                ) <= 0)
                m--;
            res.p[m++] = a[i];
        }
        int k = m;
        for (int i = a.size() - 2; i >= 0; --i) {
            while (m > k && sgn(det(res.p[m - 1] - res.p[m - 2], a[i] - res.p[m - 2])
                ) <= 0) {
                m--;
            }
            res.p[m++] = a[i];
        }
        if (m > 1) m--;
        res.p.resize(m);
        res.n = m;
        *this = res;
    }
    //
    bool isUnique(vector<Point> &v) {
        if (sgn(area()) == 0) return 0;
        for (int i = 0; i < n; ++i) {
            Line l(p[i], p[_next(i)]);
            bool flag = 0;
            for (int j = 0; j < v.size(); ++j) {
                if (l.pointOnSegInterval(v[j])) {
                    flag = 1;
                    break;
                }
            }
            if (!flag) return 0;
        }
        return 1;
    }
    // O(n)
    bool containon(const Point &a) {
        for (int sign = 0, i = 0; i < n; ++i) {
            int x = sgn(det(p[i] - a, p[_next(i)] - a));
            if (x == 0) continue; // return 0; //
            if (!sign)
                sign = x;
            else if (sign != x)
                return 0;
        }
        return 1;
    }
    // O(logn)
    bool containologn(const Point &a) {
        Point g = (p[0] + p[n / 3] + p[2.0 * n / 3]) / 3.0;
        int l = 0, r = n;
        while (l + 1 < r) {
            int m = (l + r) >> 1;
            if (sgn(det(p[l] - g, p[m] - g)) > 0) {
                if (sgn(det(p[l] - g, a - g)) >= 0 &&
                    sgn(det(p[m] - g, a - g)) < 0)
                    r = m;
                else
                    l = m;
            } else {
                if (sgn(det(p[l] - g, a - g)) < 0 &&
                    sgn(det(p[m] - g, a - g)) >= 0)

```

```

        l = m;
    else
        r = m;
    }
}
return sgn(det(p[r % n] - a, p[l] - a)) - 1;
}
//
int fir, sec; //
double diameter() {
    double mx = 0;
    if (n == 1) {
        fir = sec = 0;
        return mx;
    }
    for (int i = 0, j = 1; i < n; ++i) {
        while (sgn(det(p[_next(i)] - p[i], p[j] - p[i]) -
            det(p[_next(i)] - p[i], p[_next(j)] - p[i])) < 0) {
            j = _next(j);
        }
        double d = dis(p[i], p[j]);
        if (d > mx) {
            mx = d;
            fir = i;
            sec = j;
        }
        d = dis(p[_next(i)], p[_next(j)]);
        if (d > mx) {
            mx = d;
            fir = _next(i);
            sec = _next(j);
        }
    }
    return mx;
}
//      O(log(n)),      On,
vector<double> ang; //
bool isinitangle;
int finda(const double &x) {
    return upper_bound(ang.begin(), ang.end(), x) - ang.begin();
}
double getAngle(const Point &p) { //      [0, 2PI]
    double res = atan2(p.y, p.x); //      -PI, PI
    //      if (res < 0) res += 2 * pi; //
    if (res < -PI / 2 + eps) res += 2 * PI; //      eps
    return res;
}
void initAngle() {
    for (int i = 0; i < n; ++i) {
        ang.push_back(getAngle(p[_next(i)] - p[i]));
    }
    isinitangle = 1;
}
bool isxLine(const Line &l) {
    if (!isinitangle) initAngle();
    int i = finda(getAngle(l.t - l.s));
    int j = finda(getAngle(l.s - l.t));
    if (sgn(det(l.t - l.s, p[i] - l.s) * det(l.t - l.s, p[j] - l.s)) >= 0)
        return 0;
    return 1;
}
}
};

```

2.6 Triangle.cpp

```
#include "Line.cpp"
```

```

struct Triangle {
    Triangle(const Point &a, const Point &b, const Point &c)
        : a(a), b(b), c(c){};
    Point a, b, c;
    double getArea() { return det(b - a, c - a) * sin(arg_2(b - c, c - a)); }
    //
    Point outCenter() {
        Line u, v;
        u.s = (a + b) / 2;
        u.t.x = u.s.x - a.y + b.y;
        u.t.y = u.s.y + a.x - b.x;
        v.s = (a + c) / 2;
        v.t.x = v.s.x - a.y + c.y;
        v.t.y = v.s.y + a.x - c.x;
        return lineIntersection(u, v);
    }
    //
    Point inCenter() {
        Line u, v;
        u.s = a;
        double m = atan2(b.y - a.y, b.x - a.x);
        double n = atan2(c.y - a.y, c.x - a.x);
        u.t.x = u.s.x + cos((m + n) / 2);
        u.t.y = u.s.y + sin((m + n) / 2);
        v.s = b;
        m = atan2(a.y - b.y, a.x - b.x);
        n = atan2(c.y - b.y, c.x - b.x);
        v.t.x = v.s.x + cos((m + n) / 2);
        v.t.y = v.s.y + sin((m + n) / 2);
        return lineIntersection(u, v);
    }
    //
    Point perpenCenter() {
        Line u, v;
        u.s = c;
        u.t.x = u.s.x - a.y + b.y;
        u.t.y = u.s.y + a.x - b.x;
        v.s = b;
        v.t.x = v.s.x - a.y + c.y;
        v.t.y = v.s.y + a.x - c.x;
        return lineIntersection(u, v);
    }
    //
    //
    //
    Point baryCenter() {
        Line u((a + b) / 2, c), v((a + c) / 2, b);
        return lineIntersection(u, v);
    }
    //
    Point fermentPoint() {
        if (arg_3(a, b, c) >= 2 * PI / 3) return b;
        if (arg_3(b, a, c) >= 2 * PI / 3) return a;
        if (arg_3(a, c, b) >= 2 * PI / 3) return c;
        Point ab = (a + b) / 2, ac = (a + c) / 2;
        Point z1 = sqrt(3.0) * (a - ab), z2 = sqrt(3.0) * (a - ac);
        z1 = rotate(z1, PI / 2);
        z2 = rotate(z2, PI / 2);
        if (arg_2(z1, c - ab) < PI / 2) {
            z1.x = -z1.x;
            z1.y = -z1.y;
        }
        if (arg_2(z2, b - ac) < PI / 2) {
            z2.x = -z2.x;

```

```

        z2.y = -z2.y;
    }
    return intersection(c, ab + z1, b, ac + z2);
}
//
Point FermatPoint() {
    Point u, v;
    double step = fabs(a.x) + fabs(a.y) + fabs(b.x) + fabs(b.y) + fabs(c.x) +
        fabs(c.y);
    u = (a + b + c) / 3;
    while (step > 1e-10)
        for (int k = 0; k < 10; step /= 2, ++k)
            for (int i = -1; i <= 1; ++i) {
                for (int j = -1; j <= 1; ++j) {
                    v.x = u.x + step * i;
                    v.y = u.y + step * j;
                    if (dis(u, a) + dis(u, b) + dis(u, c) > dis(v, a) + dis(v, b)
                        + dis(v, c)) {
                        u = v;
                    }
                }
            }
    return u;
}
};

```

2.7 mygeo.cpp

```

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

```

```

#define mp make_pair
#define fi first
#define se second
#define pb push_back
typedef double db;
const db eps = 1e-6;
const db pi = acos(-1);
int sign(db k) {
    if (k > eps)
        return 1;
    else if (k < -eps)
        return -1;
    return 0;
}
int cmp(db k1, db k2) { return sign(k1 - k2); }
int inmid(db k1, db k2, db k3) {
    return sign(k1 - k3) * sign(k2 - k3) <= 0;
}
// k3 [k1, k2]
struct point {
    db x, y;
    point operator+(const point &k1) const {
        return (point){k1.x + x, k1.y + y};
    }
    point operator-(const point &k1) const {
        return (point){x - k1.x, y - k1.y};
    }
    point operator*(db k1) const { return (point){x * k1, y * k1}; }
    point operator/(db k1) const { return (point){x / k1, y / k1}; }
    int operator==(const point &k1) const {
        return cmp(x, k1.x) == 0 && cmp(y, k1.y) == 0;
    }
    //
    point turn(db k1) {
        return (point){x * cos(k1) - y * sin(k1), x * sin(k1) + y * cos(k1)};
    }
    point turn90() { return (point){-y, x}; }
}

```

```

bool operator<(const point k1) const {
    int a = cmp(x, k1.x);
    if (a == -1)
        return 1;
    else if (a == 1)
        return 0;
    else
        return cmp(y, k1.y) == -1;
}
db abs() { return sqrt(x * x + y * y); }
db abs2() { return x * x + y * y; }
db dis(point k1) { return ((*this) - k1).abs(); }
point unit() {
    db w = abs();
    return (point){x / w, y / w};
}
void scan() {
    double k1, k2;
    scanf("%lf%lf", &k1, &k2);
    x = k1;
    y = k2;
}
void print() { printf("%.11lf %.11lf\n", x, y); }
db getw() { return atan2(y, x); }
point getdel() {
    if (sign(x) == -1 || (sign(x) == 0 && sign(y) == -1))
        return (*this) * (-1);
    else
        return (*this);
}
int getP() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) == -1); }
};
int inmid(point k1, point k2, point k3) {
    return inmid(k1.x, k2.x, k3.x) && inmid(k1.y, k2.y, k3.y);
}
db cross(point k1, point k2) { return k1.x * k2.y - k1.y * k2.x; }
db dot(point k1, point k2) { return k1.x * k2.x + k1.y * k2.y; }
db rad(point k1, point k2) { return atan2(cross(k1, k2), dot(k1, k2)); }
// -pi -> pi
int compareangle(point k1, point k2) {
    return k1.getP() < k2.getP() ||
        (k1.getP() == k2.getP() && sign(cross(k1, k2)) > 0);
}
point proj(point k1, point k2, point q) { // q k1, k2
    point k = k2 - k1;
    return k1 + k * (dot(q - k1, k) / k.abs2());
}
point reflect(point k1, point k2, point q) { return proj(k1, k2, q) * 2 - q; }
int clockwise(point k1, point k2,
    point k3) { // k1 k2 k3 1 -1 0
    return sign(cross(k2 - k1, k3 - k1));
}
int checkLL(point k1, point k2, point k3,
    point k4) { // (L) (S) k1, k2 k3, k4
    return cmp(cross(k3 - k1, k4 - k1), cross(k3 - k2, k4 - k2)) != 0;
}
point getLL(point k1, point k2, point k3, point k4) {
    db w1 = cross(k1 - k3, k4 - k3), w2 = cross(k4 - k3, k2 - k3);
    return (k1 * w2 + k2 * w1) / (w1 + w2);
}
int intersect(db l1, db r1, db l2, db r2) {
    if (l1 > r1) swap(l1, r1);
    if (l2 > r2) swap(l2, r2);
    return cmp(r1, l2) != -1 && cmp(r2, l1) != -1;
}
int checkSS(point k1, point k2, point k3, point k4) {
    return intersect(k1.x, k2.x, k3.x, k4.x) &&

```

```

        intersect(k1.y, k2.y, k3.y, k4.y) &&
        sign(cross(k3 - k1, k4 - k1)) * sign(cross(k3 - k2, k4 - k2)) <= 0 &&
        sign(cross(k1 - k3, k2 - k3)) * sign(cross(k1 - k4, k2 - k4)) <= 0;
    }
    db disSP(point k1, point k2, point q) {
        point k3 = proj(k1, k2, q);
        if (inmid(k1, k2, k3))
            return q.dis(k3);
        else
            return min(q.dis(k1), q.dis(k2));
    }
    db disSS(point k1, point k2, point k3, point k4) {
        if (checkSS(k1, k2, k3, k4))
            return 0;
        else
            return min(min(disSP(k1, k2, k3), disSP(k1, k2, k4)),
                        min(disSP(k3, k4, k1), disSP(k3, k4, k2)));
    }
    int onS(point k1, point k2, point q) {
        return inmid(k1, k2, q) && sign(cross(k1 - q, k2 - k1)) == 0;
    }
    struct circle {
        point o;
        db r;
        void scan() {
            o.scan();
            scanf("%lf", &r);
        }
        int inside(point k) { return cmp(r, o.dis(k)); }
    };
    struct line {
        // p[0]->p[1]
        point p[2];
        line(point k1, point k2) {
            p[0] = k1;
            p[1] = k2;
        }
        point &operator[](int k) { return p[k]; }
        int include(point k) { return sign(cross(p[1] - p[0], k - p[0])) > 0; }
        point dir() { return p[1] - p[0]; }
        line push() { // ( ) eps
            const db eps = 1e-6;
            point delta = (p[1] - p[0]).turn90().unit() * eps;
            return {p[0] - delta, p[1] - delta};
        }
    };
    point getLL(line k1, line k2) { return getLL(k1[0], k1[1], k2[0], k2[1]); }
    int parallel(line k1, line k2) { return sign(cross(k1.dir(), k2.dir())) == 0; }
    int sameDir(line k1, line k2) {
        return parallel(k1, k2) && sign(dot(k1.dir(), k2.dir())) == 1;
    }
    int operator<(line k1, line k2) {
        if (sameDir(k1, k2)) return k2.include(k1[0]);
        return compareangle(k1.dir(), k2.dir());
    }
    int checkpos(line k1, line k2, line k3) { return k3.include(getLL(k1, k2)); }
    vector<line> getHL(
        vector<line> &L) { // , ,
        sort(L.begin(), L.end());
        deque<line> q;
        for (int i = 0; i < (int)L.size(); i++) {
            if (i && sameDir(L[i], L[i - 1])) continue;
            while (q.size() > 1 &&
                    !checkpos(q[q.size() - 2], q[q.size() - 1], L[i]))
                q.pop_back();
            while (q.size() > 1 && !checkpos(q[1], q[0], L[i])) q.pop_front();
            q.push_back(L[i]);
        }
    }

```

```

    }
    while (q.size() > 2 && !checkpos(q[q.size() - 2], q[q.size() - 1], q[0]))
        q.pop_back();
    while (q.size() > 2 && !checkpos(q[1], q[0], q[q.size() - 1]))
        q.pop_front();
    vector<line> ans;
    for (int i = 0; i < q.size(); i++) ans.push_back(q[i]);
    return ans;
}
db closepoint(vector<point> &A, int l,
               int r) { // , x
    if (r - l <= 5) {
        db ans = 1e20;
        for (int i = l; i <= r; i++)
            for (int j = i + 1; j <= r; j++) ans = min(ans, A[i].dis(A[j]));
        return ans;
    }
    int mid = (l + r) >> 1;
    db ans = min(closepoint(A, l, mid), closepoint(A, mid + 1, r));
    vector<point> B;
    for (int i = l; i <= r; i++)
        if (abs(A[i].x - A[mid].x) <= ans) B.push_back(A[i]);
    sort(B.begin(), B.end(), [](point k1, point k2) { return k1.y < k2.y; });
    for (int i = 0; i < B.size(); i++)
        for (int j = i + 1; j < B.size() && B[j].y - B[i].y < ans; j++)
            ans = min(ans, B[i].dis(B[j]));
    return ans;
}
int checkposCC(circle k1, circle k2) { //
    if (cmp(k1.r, k2.r) == -1) swap(k1, k2);
    db dis = k1.o.dis(k2.o);
    int w1 = cmp(dis, k1.r + k2.r), w2 = cmp(dis, k1.r - k2.r);
    if (w1 > 0)
        return 4;
    else if (w1 == 0)
        return 3;
    else if (w2 > 0)
        return 2;
    else if (w2 == 0)
        return 1;
    else
        return 0;
}
vector<point> getCL(circle k1, point k2,
                   point k3) { // k2->k3 ,
    point k = proj(k2, k3, k1.o);
    db d = k1.r * k1.r - (k - k1.o).abs2();
    if (sign(d) == -1) return {};
    point del = (k3 - k2).unit() * sqrt(max((db)0.0, d));
    return {k - del, k + del};
}
vector<point> getCC(circle k1,
                   circle k2) { // k1 ,
    int pd = checkposCC(k1, k2);
    if (pd == 0 || pd == 4) return {};
    db a = (k2.o - k1.o).abs2(), cosA = (k1.r * k1.r + a - k2.r * k2.r) /
                                         (2 * k1.r * sqrt(max(a, (db)0.0)));
    db b = k1.r * cosA, c = sqrt(max((db)0.0, k1.r * k1.r - b * b));
    point k = (k2.o - k1.o).unit(), m = k1.o + k * b, del = k.turn90() * c;
    return {m - del, m + del};
}
vector<point> TangentCP(circle k1, point k2) { // k1
    db a = (k2 - k1.o).abs(), b = k1.r * k1.r / a,
        c = sqrt(max((db)0.0, k1.r * k1.r - b * b));
    point k = (k2 - k1.o).unit(), m = k1.o + k * b, del = k.turn90() * c;
    return {m - del, m + del};
}

```

```

vector<line> TangentoutCC(circle k1, circle k2) {
    int pd = checkposCC(k1, k2);
    if (pd == 0) return {};
    if (pd == 1) {
        point k = getCC(k1, k2)[0];
        return {(line){k, k}};
    }
    if (cmp(k1.r, k2.r) == 0) {
        point del = (k2.o - k1.o).unit().turn90().getdel();
        return {(line){k1.o - del * k1.r, k2.o - del * k2.r},
                (line){k1.o + del * k1.r, k2.o + del * k2.r}};
    } else {
        point p = (k2.o * k1.r - k1.o * k2.r) / (k1.r - k2.r);
        vector<point> A = TangentCP(k1, p), B = TangentCP(k2, p);
        vector<line> ans;
        for (int i = 0; i < A.size(); i++) ans.push_back((line){A[i], B[i]});
        return ans;
    }
}

vector<line> TangentinCC(circle k1, circle k2) {
    int pd = checkposCC(k1, k2);
    if (pd <= 2) return {};
    if (pd == 3) {
        point k = getCC(k1, k2)[0];
        return {(line){k, k}};
    }
    point p = (k2.o * k1.r + k1.o * k2.r) / (k1.r + k2.r);
    vector<point> A = TangentCP(k1, p), B = TangentCP(k2, p);
    vector<line> ans;
    for (int i = 0; i < A.size(); i++) ans.push_back((line){A[i], B[i]});
    return ans;
}

vector<line> TangentCC(circle k1, circle k2) {
    int flag = 0;
    if (k1.r < k2.r) swap(k1, k2), flag = 1;
    vector<line> A = TangentoutCC(k1, k2), B = TangentinCC(k1, k2);
    for (line k : B) A.push_back(k);
    if (flag)
        for (line &k : A) swap(k[0], k[1]);
    return A;
}

db getarea(circle k1, point k2, point k3) {
    // k1 k2 k3 k1.o
    point k = k1.o;
    k1.o = k1.o - k;
    k2 = k2 - k;
    k3 = k3 - k;
    int pd1 = k1.inside(k2), pd2 = k1.inside(k3);
    vector<point> A = getCL(k1, k2, k3);
    if (pd1 >= 0) {
        if (pd2 >= 0) return cross(k2, k3) / 2;
        return k1.r * k1.r * rad(A[1], k3) / 2 + cross(k2, A[1]) / 2;
    } else if (pd2 >= 0) {
        return k1.r * k1.r * rad(k2, A[0]) / 2 + cross(A[0], k3) / 2;
    } else {
        int pd = cmp(k1.r, disSP(k2, k3, k1.o));
        if (pd <= 0) return k1.r * k1.r * rad(k2, k3) / 2;
        return cross(A[0], A[1]) / 2 +
            k1.r * k1.r * (rad(k2, A[0]) + rad(A[1], k3)) / 2;
    }
}

circle getcircle(point k1, point k2, point k3) {
    db a1 = k2.x - k1.x, b1 = k2.y - k1.y, c1 = (a1 * a1 + b1 * b1) / 2;
    db a2 = k3.x - k1.x, b2 = k3.y - k1.y, c2 = (a2 * a2 + b2 * b2) / 2;
    db d = a1 * b2 - a2 * b1;
    point o =
        (point){k1.x + (c1 * b2 - c2 * b1) / d, k1.y + (a1 * c2 - a2 * c1) / d};
}

```

```

    return (circle){o, k1.dis(o)};
}

circle getScircle(vector<point> A) {
    // random_shuffle(A.begin(), A.end());
    circle ans = (circle){A[0], 0};
    for (int i = 1; i < A.size(); i++)
        if (ans.inside(A[i]) == -1) {
            ans = (circle){A[i], 0};
            for (int j = 0; j < i; j++)
                if (ans.inside(A[j]) == -1) {
                    ans.o = (A[i] + A[j]) / 2;
                    ans.r = ans.o.dis(A[i]);
                    for (int k = 0; k < j; k++)
                        if (ans.inside(A[k]) == -1)
                            ans = getcircle(A[i], A[j], A[k]);
                }
        }
    return ans;
}

db area(vector<point> A) { // vector<point> ,
    db ans = 0;
    for (int i = 0; i < A.size(); i++)
        ans += cross(A[i], A[(i + 1) % A.size()]);
    return ans / 2;
}

int checkconvex(vector<point> A) {
    int n = A.size();
    A.push_back(A[0]);
    A.push_back(A[1]);
    for (int i = 0; i < n; i++)
        if (sign(cross(A[i + 1] - A[i], A[i + 2] - A[i])) == -1) return 0;
    return 1;
}

int contain(vector<point> A, point q) { // 2 1 0
    int pd = 0;
    A.push_back(A[0]);
    for (int i = 1; i < A.size(); i++) {
        point u = A[i - 1], v = A[i];
        if (onS(u, v, q)) return 1;
        if (cmp(u.y, v.y) > 0) swap(u, v);
        if (cmp(u.y, q.y) >= 0 || cmp(v.y, q.y) < 0) continue;
        if (sign(cross(u - v, q - v)) < 0) pd ^= 1;
    }
    return pd << 1;
}

vector<point> ConvexHull(vector<point> A,
    int flag = 1) { // flag=0 flag=1
    int n = A.size();
    vector<point> ans(n * 2);
    sort(A.begin(), A.end());
    int now = -1;
    for (int i = 0; i < A.size(); i++) {
        while (now > 0 &&
            sign(cross(ans[now] - ans[now - 1], A[i] - ans[now - 1])) < flag)
            now--;
        ans[++now] = A[i];
    }
    int pre = now;
    for (int i = n - 2; i >= 0; i--) {
        while (now > pre &&
            sign(cross(ans[now] - ans[now - 1], A[i] - ans[now - 1])) < flag)
            now--;
        ans[++now] = A[i];
    }
    ans.resize(now);
    return ans;
}

```



```

db convexDiameter(vector<point> A) {
    int now = 0, n = A.size();
    db ans = 0;
    for (int i = 0; i < A.size(); i++) {
        now = max(now, i);
        while (1) {
            db k1 = A[i].dis(A[now % n]), k2 = A[i].dis(A[(now + 1) % n]);
            ans = max(ans, max(k1, k2));
            if (k2 > k1)
                now++;
            else
                break;
        }
    }
    return ans;
}

vector<point> convexcut(vector<point> A, point k1, point k2) {
    // k1, k2, p
    int n = A.size();
    A.push_back(A[0]);
    vector<point> ans;
    for (int i = 0; i < n; i++) {
        int w1 = clockwise(k1, k2, A[i]), w2 = clockwise(k1, k2, A[i + 1]);
        if (w1 >= 0) ans.push_back(A[i]);
        if (w1 * w2 < 0) ans.push_back(getLL(k1, k2, A[i], A[i + 1]));
    }
    return ans;
}

int checkPoS(vector<point> A, point k1, point k2) {
    // A ( ) k1->k2
    struct ins {
        point m, u, v;
        int operator<(const ins &k) const { return m < k.m; }
    };
    vector<ins> B;
    // if (contain(A, k1) == 2 || contain(A, k2) == 2) return 1;
    vector<point> poly = A;
    A.push_back(A[0]);
    for (int i = 1; i < A.size(); i++)
        if (checkLL(A[i - 1], A[i], k1, k2)) {
            point m = getLL(A[i - 1], A[i], k1, k2);
            if (inmid(A[i - 1], A[i], m) /*&& inmid(k1, k2, m)*/)
                B.push_back((ins){m, A[i - 1], A[i]});
        }
    if (B.size() == 0) return 0;
    sort(B.begin(), B.end());
    int now = 1;
    while (now < B.size() && B[now].m == B[0].m) now++;
    if (now == B.size()) return 0;
    int flag = contain(poly, (B[0].m + B[now].m) / 2);
    if (flag == 2) return 1;
    point d = B[now].m - B[0].m;
    for (int i = now; i < B.size(); i++) {
        if (!(B[i].m == B[i - 1].m) && flag == 2) return 1;
        int tag = sign(cross(B[i].v - B[i].u, B[i].m + d - B[i].u));
        if (B[i].m == B[i].u || B[i].m == B[i].v)
            flag += tag;
        else
            flag += tag * 2;
    }
    // return 0;
    return flag == 2;
}

int checkinp(point r, point l, point m) {
    if (compareangle(l, r)) {
        return compareangle(l, m) && compareangle(m, r);
    }
}

```

```

        return compareangle(l, m) || compareangle(m, r);
    }

int checkPosFast(vector<point> A, point k1,
    point k2) { //
    if (contain(A, k1) == 2 || contain(A, k2) == 2) return 1;
    if (k1 == k2) return 0;
    A.push_back(A[0]);
    A.push_back(A[1]);
    for (int i = 1; i + 1 < A.size(); i++)
        if (checkLL(A[i - 1], A[i], k1, k2)) {
            point now = getLL(A[i - 1], A[i], k1, k2);
            if (inmid(A[i - 1], A[i], now) == 0 || inmid(k1, k2, now) == 0)
                continue;
            if (now == A[i]) {
                if (A[i] == k2) continue;
                point pre = A[i - 1], ne = A[i + 1];
                if (checkinp(pre - now, ne - now, k2 - now)) return 1;
            } else if (now == k1) {
                if (k1 == A[i - 1] || k1 == A[i]) continue;
                if (checkinp(A[i - 1] - k1, A[i] - k1, k2 - k1)) return 1;
            } else if (now == k2 || now == A[i - 1])
                continue;
            else
                return 1;
        }
    return 0;
}

//
//
void getUDP(vector<point> A, vector<point> &U, vector<point> &D) {
    db l = 1e100, r = -1e100;
    for (int i = 0; i < A.size(); i++) l = min(l, A[i].x), r = max(r, A[i].x);
    int wherel, wherer;
    for (int i = 0; i < A.size(); i++)
        if (cmp(A[i].x, l) == 0) wherel = i;
    for (int i = A.size(); i; i--)
        if (cmp(A[i - 1].x, r) == 0) wherer = i - 1;
    U.clear();
    D.clear();
    int now = wherel;
    while (1) {
        D.push_back(A[now]);
        if (now == wherer) break;
        now++;
        if (now >= A.size()) now = 0;
    }
    now = wherel;
    while (1) {
        U.push_back(A[now]);
        if (now == wherer) break;
        now--;
        if (now < 0) now = A.size() - 1;
    }
}

// 3, 2, 1, 0
int containCoP(const vector<point> &U, const vector<point> &D, point k) {
    db lx = U[0].x, rx = U[U.size() - 1].x;
    if (k == U[0] || k == U[U.size() - 1]) return 1;
    if (cmp(k.x, lx) == -1 || cmp(k.x, rx) == 1) return 0;
    int wherel =
        lower_bound(U.begin(), U.end(), (point){k.x, -1e100}) - U.begin();
    int where2 =
        lower_bound(D.begin(), D.end(), (point){k.x, -1e100}) - D.begin();
    int w1 = clockwise(U[wherel - 1], U[wherel], k),
        w2 = clockwise(D[where2 - 1], D[where2], k);
    if (w1 == 1 || w2 == -1)
        return 0;
}

```

```

else if (w1 == 0 || w2 == 0)
    return 1;
return 2;
}
// d
pair<point, point> getTangentCow(const vector<point> &U, const vector<point> &D,
    point d) {
    if (sign(d.x) < 0 || (sign(d.x) == 0 && sign(d.y) < 0)) d = d * (-1);
    point whereU, whereD;
    if (sign(d.x) == 0) return mp(U[0], U[U.size() - 1]);
    int l = 0, r = U.size() - 1, ans = 0;
    while (l < r) {
        int mid = (l + r) >> 1;
        if (sign(cross(U[mid + 1] - U[mid], d)) <= 0)
            l = mid + 1, ans = mid + 1;
        else
            r = mid;
    }
    whereU = U[ans];
    l = 0, r = D.size() - 1, ans = 0;
    while (l < r) {
        int mid = (l + r) >> 1;
        if (sign(cross(D[mid + 1] - D[mid], d)) >= 0)
            l = mid + 1, ans = mid + 1;
        else
            r = mid;
    }
    whereD = D[ans];
    return mp(whereU, whereD);
}
// contain,
pair<point, point> getTangentCoP(const vector<point> &U, const vector<point> &D,
    point k) {
    db lx = U[0].x, rx = U[U.size() - 1].x;
    if (k.x < lx) {
        int l = 0, r = U.size() - 1, ans = U.size() - 1;
        while (l < r) {
            int mid = (l + r) >> 1;
            if (clockwise(k, U[mid], U[mid + 1]) == 1)
                l = mid + 1;
            else
                ans = mid, r = mid;
        }
        point w1 = U[ans];
        l = 0, r = D.size() - 1, ans = D.size() - 1;
        while (l < r) {
            int mid = (l + r) >> 1;
            if (clockwise(k, D[mid], D[mid + 1]) == -1)
                l = mid + 1;
            else
                ans = mid, r = mid;
        }
        point w2 = D[ans];
        return mp(w1, w2);
    } else if (k.x > rx) {
        int l = 1, r = U.size(), ans = 0;
        while (l < r) {
            int mid = (l + r) >> 1;
            if (clockwise(k, U[mid], U[mid - 1]) == -1)
                r = mid;
            else
                ans = mid, l = mid + 1;
        }
        point w1 = U[ans];
        l = 1, r = D.size(), ans = 0;
        while (l < r) {
            int mid = (l + r) >> 1;

```

```

            if (clockwise(k, D[mid], D[mid - 1]) == 1)
                r = mid;
            else
                ans = mid, l = mid + 1;
        }
        point w2 = D[ans];
        return mp(w2, w1);
    } else {
        int where1 =
            lower_bound(U.begin(), U.end(), (point){k.x, -1e100}) - U.begin();
        int where2 =
            lower_bound(D.begin(), D.end(), (point){k.x, -1e100}) - D.begin();
        if ((k.x == lx && k.y > U[0].y) ||
            (where1 && clockwise(U[where1 - 1], U[where1], k) == 1)) {
            int l = 1, r = where1 + 1, ans = 0;
            while (l < r) {
                int mid = (l + r) >> 1;
                if (clockwise(k, U[mid], U[mid - 1]) == 1)
                    ans = mid, l = mid + 1;
                else
                    r = mid;
            }
            point w1 = U[ans];
            l = where1, r = U.size() - 1, ans = U.size() - 1;
            while (l < r) {
                int mid = (l + r) >> 1;
                if (clockwise(k, U[mid], U[mid + 1]) == 1)
                    l = mid + 1;
                else
                    ans = mid, r = mid;
            }
            point w2 = U[ans];
            return mp(w2, w1);
        } else {
            int l = 1, r = where2 + 1, ans = 0;
            while (l < r) {
                int mid = (l + r) >> 1;
                if (clockwise(k, D[mid], D[mid - 1]) == -1)
                    ans = mid, l = mid + 1;
                else
                    r = mid;
            }
            point w1 = D[ans];
            l = where2, r = D.size() - 1, ans = D.size() - 1;
            while (l < r) {
                int mid = (l + r) >> 1;
                if (clockwise(k, D[mid], D[mid + 1]) == -1)
                    l = mid + 1;
                else
                    ans = mid, r = mid;
            }
            point w2 = D[ans];
            return mp(w1, w2);
        }
    }
}
}
struct P3 {
    db x, y, z;
    P3 operator+(P3 k1) { return (P3){x + k1.x, y + k1.y, z + k1.z}; }
    P3 operator-(P3 k1) { return (P3){x - k1.x, y - k1.y, z - k1.z}; }
    P3 operator*(db k1) { return (P3){x * k1, y * k1, z * k1}; }
    P3 operator/(db k1) { return (P3){x / k1, y / k1, z / k1}; }
    db abs2() { return x * x + y * y + z * z; }
    db abs() { return sqrt(x * x + y * y + z * z); }
    P3 unit() { return (*this) / abs(); }
    int operator<(const P3 k1) const {
        if (cmp(x, k1.x) != 0) return x < k1.x;

```

```

        if (cmp(y, k1.y) != 0) return y < k1.y;
        return cmp(z, k1.z) == -1;
    }
    int operator==(const P3 k1) {
        return cmp(x, k1.x) == 0 && cmp(y, k1.y) == 0 && cmp(z, k1.z) == 0;
    }
    void scan() {
        double k1, k2, k3;
        scanf("%lf%lf%lf", &k1, &k2, &k3);
        x = k1;
        y = k2;
        z = k3;
    }
};
P3 cross(P3 k1, P3 k2) {
    return (P3){k1.y * k2.z - k1.z * k2.y, k1.z * k2.x - k1.x * k2.z,
        k1.x * k2.y - k1.y * k2.x};
}
db dot(P3 k1, P3 k2) { return k1.x * k2.x + k1.y * k2.y + k1.z * k2.z; }
// p=(3,4,5),l=(13,19,21),theta=85 ans=(2.83,4.62,1.77)
P3 turn3D(db k1, P3 l, P3 p) {
    l = l.unit();
    P3 ans;
    db c = cos(k1), s = sin(k1);
    ans.x = p.x * (l.x * l.x * (1 - c) + c) +
        p.y * (l.x * l.y * (1 - c) - l.z * s) +
        p.z * (l.x * l.z * (1 - c) + l.y * s);
    ans.y = p.x * (l.x * l.y * (1 - c) + l.z * s) +
        p.y * (l.y * l.y * (1 - c) + c) +
        p.z * (l.y * l.z * (1 - c) - l.x * s);
    ans.z = p.x * (l.x * l.z * (1 - c) - l.y * s) +
        p.y * (l.y * l.z * (1 - c) + l.x * s) +
        p.z * (l.x * l.x * (1 - c) + c);
    return ans;
}
typedef vector<P3> VP;
typedef vector<VP> VVP;
db Acos(db x) { return acos(max(-(db)1, min(x, (db)1))); }
// , 1
db Odist(P3 a, P3 b) {
    db r = Acos(dot(a, b));
    return r;
}
db r;
P3 rnd;
vector<db> solve(db a, db b, db c) {
    db r = sqrt(a * a + b * b), th = atan2(b, a);
    if (cmp(c, -r) == -1)
        return {0};
    else if (cmp(r, c) <= 0)
        return {1};
    else {
        db tr = pi - Acos(c / r);
        return {th + pi - tr, th + pi + tr};
    }
}
vector<db> jiao(P3 a, P3 b) {
    // dot(rd+x*cos(t)+y*sin(t),b) >= cos(r)
    if (cmp(Odist(a, b), 2 * r) > 0) return {0};
    P3 rd = a * cos(r), z = a.unit(), y = cross(z, rnd).unit(),
        x = cross(y, z).unit();
    vector<db> ret = solve(-(dot(x, b) * sin(r)), -(dot(y, b) * sin(r)),
        -(cos(r) - dot(rd, b)));
    return ret;
}
}
db norm(db x, db l = 0, db r = 2 * pi) { // change x into [l,r)
    while (cmp(x, l) == -1) x += (r - l);
}

```

```

    while (cmp(x, r) >= 0) x -= (r - l);
    return x;
}
db disLP(P3 k1, P3 k2, P3 q) {
    return (cross(k2 - k1, q - k1)).abs() / (k2 - k1).abs();
}
db disLL(P3 k1, P3 k2, P3 k3, P3 k4) {
    P3 dir = cross(k2 - k1, k4 - k3);
    if (sign(dir.abs()) == 0) return disLP(k1, k2, k3);
    return fabs(dot(dir.unit(), k1 - k2));
}
VP getFL(P3 p, P3 dir, P3 k1, P3 k2) {
    db a = dot(k2 - p, dir), b = dot(k1 - p, dir), d = a - b;
    if (sign(fabs(d)) == 0) return {};
    return {(k1 * a - k2 * b) / d};
}
VP getFF(P3 p1, P3 dir1, P3 p2, P3 dir2) { //
    P3 e = cross(dir1, dir2), v = cross(dir1, e);
    db d = dot(dir2, v);
    if (sign(abs(d)) == 0) return {};
    P3 q = p1 + v * dot(dir2, p2 - p1) / d;
    return {q, q + e};
}
// 3D Convex Hull Template
db getV(P3 k1, P3 k2, P3 k3, P3 k4) { // get the Volume
    return dot(cross(k2 - k1, k3 - k1), k4 - k1);
}
db rand_db() { return 1.0 * rand() / RAND_MAX; }
VP convexHull2D(VP A, P3 dir) {
    P3 x = {(db)rand(), (db)rand(), (db)rand()};
    x = x.unit();
    x = cross(x, dir).unit();
    P3 y = cross(x, dir).unit();
    P3 vec = dir.unit() * dot(A[0], dir);
    vector<point> B;
    for (int i = 0; i < A.size(); i++)
        B.push_back((point){dot(A[i], x), dot(A[i], y)});
    B = ConvexHull(B);
    A.clear();
    for (int i = 0; i < B.size(); i++)
        A.push_back(x * B[i].x + y * B[i].y + vec);
    return A;
}
}
namespace CH3 {
    VVP ret;
    set<pair<int, int> > e;
    int n;
    VP p, q;
    void wrap(int a, int b) {
        if (e.find({a, b}) == e.end()) {
            int c = -1;
            for (int i = 0; i < n; i++)
                if (i != a && i != b) {
                    if (c == -1 || sign(getV(q[c], q[a], q[b], q[i])) > 0) c = i;
                }
            if (c != -1) {
                ret.push_back({p[a], p[b], p[c]});
                e.insert({a, b});
                e.insert({b, c});
                e.insert({c, a});
                wrap(c, b);
                wrap(a, c);
            }
        }
    }
}
VVP ConvexHull3D(VP _p) {
    p = q = _p;
}

```

```

n = p.size();
ret.clear();
e.clear();
for (auto &i : q)
    i = i + (P3){rand_db() * 1e-4, rand_db() * 1e-4, rand_db() * 1e-4};
for (int i = 1; i < n; i++)
    if (q[i].x < q[0].x) swap(p[0], p[i]), swap(q[0], q[i]);
for (int i = 2; i < n; i++)
    if ((q[i].x - q[0].x) * (q[1].y - q[0].y) >
        (q[i].y - q[0].y) * (q[1].x - q[0].x))
        swap(q[1], q[i]), swap(p[1], p[i]);
wrap(0, 1);
return ret;
}
} // namespace CH3
VVP reduceCH(VVP A) {
    VVP ret;
    map<P3, VP> M;
    for (VP nowF : A) {
        P3 dir = cross(nowF[1] - nowF[0], nowF[2] - nowF[0]).unit();
        for (P3 k1 : nowF) M[dir].pb(k1);
    }
    for (pair<P3, VP> nowF : M) ret.pb(convexHull2D(nowF.se, nowF.fi));
    return ret;
}
// ( , )
pair<P3, P3> getF(VP F) {
    return mp(F[0], cross(F[1] - F[0], F[2] - F[0]).unit());
}
// 3D Cut dot(dir, x-p) >= 0
VVP ConvexCut3D(VVP A, P3 p, P3 dir) {
    VVP ret;
    VP sec;
    for (VP nowF : A) {
        int n = nowF.size();
        VP ans;
        int dif = 0;
        for (int i = 0; i < n; i++) {
            int d1 = sign(dot(dir, nowF[i] - p));
            int d2 = sign(dot(dir, nowF[(i + 1) % n] - p));
            if (d1 >= 0) ans.pb(nowF[i]);
            if (d1 * d2 < 0) {
                P3 q = getFL(p, dir, nowF[i], nowF[(i + 1) % n])[0];
                ans.push_back(q);
                sec.push_back(q);
            }
            if (d1 == 0)
                sec.push_back(nowF[i]);
            else
                dif = 1;
            dif |= (sign(dot(dir, cross(nowF[(i + 1) % n] - nowF[i],
                nowF[(i + 1) % n] - nowF[i]))) == -1);
        }
        if (ans.size() > 0 && dif) ret.push_back(ans);
    }
    if (sec.size() > 0) ret.push_back(convexHull2D(sec, dir));
    return ret;
}
db vol(VVP A) {
    if (A.size() == 0) return 0;
    P3 p = A[0][0];
    db ans = 0;
    for (VP nowF : A)
        for (int i = 2; i < nowF.size(); i++)
            ans += abs(getV(p, nowF[0], nowF[i - 1], nowF[i]));
    return ans / 6;
}

```

```

VVP init(db INF) {
    VVP pss(6, VP(4));
    pss[0][0] = pss[1][0] = pss[2][0] = {-INF, -INF, -INF};
    pss[0][3] = pss[1][1] = pss[5][2] = {-INF, -INF, INF};
    pss[0][1] = pss[2][3] = pss[4][2] = {-INF, INF, -INF};
    pss[0][2] = pss[5][3] = pss[4][1] = {-INF, INF, INF};
    pss[1][3] = pss[2][1] = pss[3][2] = {INF, -INF, -INF};
    pss[1][2] = pss[5][1] = pss[3][3] = {INF, -INF, INF};
    pss[2][2] = pss[4][3] = pss[3][1] = {INF, INF, -INF};
    pss[5][0] = pss[4][0] = pss[3][0] = {INF, INF, INF};
    return pss;
}

```

3 Graph

3.1 2sat.cpp

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

```

using namespace std;
using ll = long long;

struct TwoSat {
    int n;
    vector<vector<int>>> G;
    vector<bool> ans;
    TwoSat(int n) : n(n), G(2 * n), ans(n) {}
    void addClause(int u, bool f, int v, bool g) {
        G[2 * u + !f].push_back(2 * v + g);
        G[2 * v + !g].push_back(2 * u + f);
    }
    bool satisfiable() {
        vector<int> id(2 * n, -1), dfn(2 * n, -1), low(2 * n, -1);
        vector<int> stk;
        int now = 0, cnt = 0;
        function<void(int)> tarjan = [&](int u) {
            stk.push_back(u);
            dfn[u] = low[u] = now++;
            for (auto v : G[u]) {
                if (dfn[v] == -1) {
                    tarjan(v);
                    low[u] = min(low[u], low[v]);
                } else if (id[v] == -1) {
                    low[u] = min(low[u], dfn[v]);
                }
            }
            if (dfn[u] == low[u]) {
                int v;
                do {
                    v = stk.back();
                    stk.pop_back();
                    id[v] = cnt;
                } while (v != u);
                ++cnt;
            }
        };
        for (int i = 0; i < 2 * n; ++i) if (dfn[i] == -1) tarjan(i);
        for (int i = 0; i < n; ++i) {
            if (id[2 * i] == id[2 * i + 1]) return false;
            ans[i] = id[2 * i] > id[2 * i + 1];
        }
        return true;
    }
    vector<bool> answer() { return ans; }
};

```

3.2 Graph.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

template <typename T>
class graph {
public:
    struct edge {
        int from;
        int to;
        T cost;
    };

    vector<edge> edges;
    vector<vector<int>>> g;
    int n;

    graph(int _n) : n(_n) { g.resize(n); }

    virtual int add(int from, int to, T cost) = 0;
};

template <typename T>
class forest : public graph<T> {
public:
    using graph<T>::edges;
    using graph<T>::g;
    using graph<T>::n;

    forest(int _n) : graph<T>(_n) {}

    int add(int from, int to, T cost = 1) {
        assert(0 <= from && from < n && 0 <= to && to < n);
        int id = (int)edges.size();
        assert(id < n - 1);
        g[from].push_back(id);
        g[to].push_back(id);
        edges.push_back({from, to, cost});
        return id;
    }
};

template <typename T>
class dfs_forest : public forest<T> {
public:
    using forest<T>::edges;
    using forest<T>::g;
    using forest<T>::n;

    vector<int> pv;
    vector<int> pe;
    vector<int> order;
    vector<int> pos;
    vector<int> end;
    vector<int> sz;
    vector<int> root;
    vector<int> depth;
    vector<T> dist;

    dfs_forest(int _n) : forest<T>(_n) {}

    void init() {
        pv = vector<int>(n, -1);
        pe = vector<int>(n, -1);
        order.clear();
        pos = vector<int>(n, -1);

        end = vector<int>(n, -1);
        sz = vector<int>(n, 0);
        root = vector<int>(n, -1);
        depth = vector<int>(n, -1);
        dist = vector<T>(n);
    }

    void clear() {
        pv.clear();
        pe.clear();
        order.clear();
        pos.clear();
        end.clear();
        sz.clear();
        root.clear();
        depth.clear();
        dist.clear();
    }

private:
    void do_dfs(int v) {
        pos[v] = (int)order.size();
        order.push_back(v);
        sz[v] = 1;
        for (int id : g[v]) {
            if (id == pe[v]) {
                continue;
            }
            auto &e = edges[id];
            int to = e.from ^ e.to ^ v;
            depth[to] = depth[v] + 1;
            dist[to] = dist[v] + e.cost;
            pv[to] = v;
            pe[to] = id;
            root[to] = (root[v] != -1 ? root[v] : to);
            do_dfs(to);
            sz[v] += sz[to];
        }
        end[v] = (int)order.size() - 1;
    }

    void do_dfs_from(int v) {
        depth[v] = 0;
        dist[v] = T{};
        root[v] = v;
        pv[v] = pe[v] = -1;
        do_dfs(v);
    }

public:
    void dfs(int v, bool clear_order = true) {
        if (pv.empty()) {
            init();
        } else {
            if (clear_order) {
                order.clear();
            }
        }
        do_dfs_from(v);
    }

    void dfs_all() {
        init();
        for (int v = 0; v < n; v++) {
            if (depth[v] == -1) {
                do_dfs_from(v);
            }
        }
    }
};

```

```

    }
    assert((int)order.size() == n);
}
};

template <typename T>
class lca_forest : public dfs_forest<T> {
public:
    using dfs_forest<T>::edges;
    using dfs_forest<T>::g;
    using dfs_forest<T>::n;
    using dfs_forest<T>::pv;
    using dfs_forest<T>::pos;
    using dfs_forest<T>::end;
    using dfs_forest<T>::depth;

    int h;
    vector<vector<int>>> pr;

    lca_forest(int _n) : dfs_forest<T>(_n) {}

    inline void build_lca() {
        assert(!pv.empty());
        int max_depth = 0;
        for (int i = 0; i < n; i++) {
            max_depth = max(max_depth, depth[i]);
        }
        h = 1;
        while ((1 << h) <= max_depth) {
            h++;
        }
        pr.resize(n);
        for (int i = 0; i < n; i++) {
            pr[i].resize(h);
            pr[i][0] = pv[i];
        }
        for (int j = 1; j < h; j++) {
            for (int i = 0; i < n; i++) {
                pr[i][j] = (pr[i][j - 1] == -1 ? -1 : pr[pr[i][j - 1]][j - 1]);
            }
        }
    }

    inline bool anc(int x, int y) {
        return (pos[x] <= pos[y] && end[y] <= end[x]);
    }

    inline int go_up(int x, int up) {
        assert(!pr.empty());
        up = min(up, (1 << h) - 1);
        for (int j = h - 1; j >= 0; j--) {
            if (up & (1 << j)) {
                x = pr[x][j];
                if (x == -1) {
                    break;
                }
            }
        }
        return x;
    }

    inline int lca(int x, int y) {
        assert(!pr.empty());
        if (anc(x, y)) {
            return x;
        }
        if (anc(y, x)) {

```

```

            return y;
        }
        for (int j = h - 1; j >= 0; j--) {
            if (pr[x][j] != -1 && !anc(pr[x][j], y)) {
                x = pr[x][j];
            }
        }
        return pr[x][0];
    }
};

3.3 MaxAssignment.cpp

#include <bits/stdc++.h>

using i64 = long long;

template<class T>
struct MaxAssignment {
public:
    T solve(int nx, int ny, std::vector<std::vector<T>>> a) {
        assert(0 <= nx && nx <= ny);
        assert(int(a.size()) == nx);
        for (int i = 0; i < nx; ++i) {
            assert(int(a[i].size()) == ny);
            for (auto x : a[i])
                assert(x >= 0);
        }

        auto update = [&](int x) {
            for (int y = 0; y < ny; ++y) {
                if (lx[x] + ly[y] - a[x][y] < slack[y]) {
                    slack[y] = lx[x] + ly[y] - a[x][y];
                    slackx[y] = x;
                }
            }
        };

        costs.resize(nx + 1);
        costs[0] = 0;
        lx.assign(nx, std::numeric_limits<T>::max());
        ly.assign(ny, 0);
        xy.assign(nx, -1);
        yx.assign(ny, -1);
        slackx.resize(ny);
        for (int cur = 0; cur < nx; ++cur) {
            std::queue<int> que;
            visx.assign(nx, false);
            visy.assign(ny, false);
            slack.assign(ny, std::numeric_limits<T>::max());
            p.assign(nx, -1);

            for (int x = 0; x < nx; ++x) {
                if (xy[x] == -1) {
                    que.push(x);
                    visx[x] = true;
                    update(x);
                }
            }

            int ex, ey;
            bool found = false;
            while (!found) {
                while (!que.empty() && !found) {
                    auto x = que.front();
                    que.pop();
                    for (int y = 0; y < ny; ++y) {

```

```

        if (a[x][y] == lx[x] + ly[y] && !visy[y]) {
            if (yx[y] == -1) {
                ex = x;
                ey = y;
                found = true;
                break;
            }
            que.push(yx[y]);
            p[yx[y]] = x;
            visy[y] = visx[yx[y]] = true;
            update(yx[y]);
        }
    }
    if (found)
        break;

    T delta = std::numeric_limits<T>::max();
    for (int y = 0; y < ny; ++y)
        if (!visy[y])
            delta = std::min(delta, slack[y]);
    for (int x = 0; x < nx; ++x)
        if (visx[x])
            lx[x] -= delta;
    for (int y = 0; y < ny; ++y) {
        if (visy[y]) {
            ly[y] += delta;
        } else {
            slack[y] -= delta;
        }
    }
    for (int y = 0; y < ny; ++y) {
        if (!visy[y] && slack[y] == 0) {
            if (yx[y] == -1) {
                ex = slackx[y];
                ey = y;
                found = true;
                break;
            }
            que.push(yx[y]);
            p[yx[y]] = slackx[y];
            visy[y] = visx[yx[y]] = true;
            update(yx[y]);
        }
    }
}

costs[cur + 1] = costs[cur];
for (int x = ex, y = ey, ty; x != -1; x = p[x], y = ty) {
    costs[cur + 1] += a[x][y];
    if (xy[x] != -1)
        costs[cur + 1] -= a[x][xy[x]];
    ty = xy[x];
    xy[x] = y;
    yx[y] = x;
}
}
return costs[nx];
}

std::vector<int> assignment() {
    return xy;
}

std::pair<std::vector<T>, std::vector<T>> labels() {
    return std::make_pair(lx, ly);
}

std::vector<T> weights() {
    return costs;
}

```

```

    }
private:
    std::vector<T> lx, ly, slack, costs;
    std::vector<int> xy, yx, p, slackx;
    std::vector<bool> visx, visy;
};

constexpr i64 inf = 1E12;

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    int n;
    std::cin >> n;

    std::vector cost(150, std::vector<i64>(150));
    for (int i = 0; i < n; i++) {
        int a, b, c;
        std::cin >> a >> b >> c;
        a--;
        b--;
        cost[a][b] = std::max(cost[a][b], inf + c);
    }

    MaxAssignment<i64> m;
    m.solve(150, 150, cost);

    int k = 0;
    auto ans = m.weights();
    while (k < 150 && ans[k + 1] >= inf * (k + 1)) {
        k++;
    }

    std::cout << k << "\n";
    for (int i = 1; i <= k; i++) {
        std::cout << ans[i] - inf * i << "\n";
    }

    return 0;
}

//test problem: https://atcoder.jp/contests/abc247/tasks/abc247_g

```

3.4 Mincost.cpp

```

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

using ll = long long;

template <typename cap_t, typename cost_t>
struct Mincost {
    static constexpr cost_t INF = numeric_limits<cost_t>::max();
    int n;
    struct Edge {
        int to;
        cap_t cap;
        cost_t cost;
        Edge(int to, cap_t cap, cost_t cost) : to(to), cap(cap), cost(cost) {}
    };
    vector<Edge> e;
    vector<vector<int>> g;
    vector<int> cur, pre;
    vector<bool> vis;
    vector<cost_t> dis;
    Mincost(int n) : n(n), g(n), vis(n) {}
}

```

```

void addEdge(int u, int v, cap_t c, cost_t w) {
    g[u].push_back(e.size());
    e.emplace_back(v, c, w);
    g[v].push_back(e.size());
    e.emplace_back(u, 0, -w);
}

bool spfa(int s, int t) {
    pre.assign(n, -1);
    dis.assign(n, INF);
    queue<int> que;
    que.push(s);
    dis[s] = 0;
    while (!que.empty()) {
        int u = que.front();
        que.pop();
        vis[u] = false;
        for (auto j : g[u]) {
            auto [v, c, w] = e[j];
            if (c > 0 && dis[v] > dis[u] + w) {
                dis[v] = dis[u] + w;
                pre[v] = j;
                if (!vis[v]) {
                    que.push(v);
                    vis[v] = true;
                }
            }
        }
    }
    return dis[t] != INF;
}

pair<cap_t, cost_t> dfs(int u, int t, cap_t f) {
    if (u == t) return {f, 0};
    vis[u] = true;
    cap_t r = f;
    cost_t p = 0;
    for (int &i = cur[u]; i < int(g[u].size()); ++i) {
        int j = g[u][i];
        auto [v, c, w] = e[j];
        if (!vis[v] && c > 0 && dis[v] == dis[u] + w) {
            auto a = dfs(v, t, min(c, r));
            e[j].cap -= a.first;
            e[j ^ 1].cap += a.first;
            r -= a.first;
            p += a.first * w + a.second;
            if (r == 0) break;
        }
    }
    vis[u] = false;
    return {f - r, p};
}

void augment(int s, int t, pair<cap_t, cost_t> &ans) {
    int p = t;
    cap_t _f = INF;
    while (pre[p] != -1) {
        _f = min(_f, e[pre[p]].cap);
        p = e[pre[p] ^ 1].to;
    }
    ans.first += _f;
    ans.second += _f * dis[t];
    p = t;
    while (pre[p] != -1) {
        e[pre[p]].cap -= _f;
        e[pre[p] ^ 1].cap += _f;
        p = e[pre[p] ^ 1].to;
    }
}

// select dfs or augment

```

```

// dfs() can multiple augment
// augment() can augment a minimum cost flow
pair<cap_t, cost_t> maxFlowMinCost(int s, int t) {
    pair<cap_t, cost_t> ans = {0, 0};
    while (spfa(s, t)) {
        cur.assign(n, 0);
        auto res = dfs(s, t, INF);
        ans.first += res.first;
        ans.second += res.second;

        // augment(s, t, ans);
    }
    return ans;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    Mincost<ll, ll> flow(n);
    const int source = 0, sink = n - 1;

    for (int i = 0; i < m; ++i) {
        int u, v;
        ll c, w;
        cin >> u >> v >> c >> w;
        u--, v--;
        flow.addEdge(u, v, c, w);
    }

    auto ans = flow.maxFlowMinCost(source, sink);
    cout << ans.first << " " << ans.second << "\n";

    return 0;
}

```

// test problem: <https://loj.ac/p/102>

3.5 Tree.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

struct Tree {
    vector<int> sz, top, dep, parent, in, out;
    int cur;
    vector<vector<int>> e;
    Tree(int n) : sz(n), top(n), dep(n), parent(n, -1), in(n), out(0), cur(0), e(n) {}
    void addEdge(int u, int v) {
        e[u].push_back(v);
        e[v].push_back(u);
    }
    void init() {
        dfsSz(0);
        dfsHLD(0);
    }
    void dfsSz(int u) {
        if (parent[u] != -1) {
            e[u].erase(find(e[u].begin(), e[u].end(), parent[u]));
        }
    }
}

```



```

    sz[u] = 1;
    for (int &v : e[u]) {
        parent[v] = u;
        dep[v] = dep[u] + 1;
        dfsSz(v);
        sz[u] += sz[v];
        if (sz[v] > sz[e[u][0]]) {
            swap(v, e[u][0]);
        }
    }
}

void dfsHLD(int u) {
    in[u] = cur++;
    for (int v : e[u]) {
        top[v] = (v == e[u][0] ? top[u] : v);
        dfsHLD(v);
    }
    out[u] = cur;
}

int lca(int u, int v) {
    while (top[u] != top[v]) {
        if (dep[top[u]] < dep[top[v]]) {
            swap(u, v);
        }
        u = parent[top[u]];
    }
    return dep[u] < dep[v] ? u : v;
}
};

```

3.6 dijkstra.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m, s;
    cin >> n >> m >> s; s--;
    vector<vector<pair<int, int>>> g(n);
    vector<int> w(m);
    for (int i = 0; i < m; ++i) {
        int u, v;
        cin >> u >> v >> w[i];
        u--, v--;
        g[u].emplace_back(v, i);
    }

    auto dijkstra = [&]() {
        vector<int> dis(n, -1);
        priority_queue<pair<int, int>> h;
        h.emplace(0, s);
        while (!h.empty()) {
            auto [d, u] = h.top();
            h.pop();
            if (dis[u] != -1) continue;
            dis[u] = -d;
            for (auto [v, j] : g[u]) {
                h.emplace(d - w[j], v);
            }
        }
        return dis;
    };
};

```

```

    auto dis = dijkstra();
    for (int i = 0; i < n; ++i) {
        cout << dis[i] << " \n"[i == n - 1];
    }

    return 0;
}

```

// test problem: <https://www.luogu.com.cn/problem/P4779>

3.7 dinic.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

template<class cap_t>
struct Flow {
    static constexpr cap_t INF = numeric_limits<cap_t>::max();
    int n;
    struct Edge {
        int to;
        cap_t cap;
        Edge(int to, cap_t cap) : to(to), cap(cap) {}
    };
    vector<Edge> e;
    vector<vector<int>> g;
    vector<int> cur, h;
    Flow(int n) : n(n), g(n) {}
    bool bfs(int s, int t) {
        h.assign(n, -1);
        queue<int> que;
        h[s] = 0;
        que.push(s);
        while (!que.empty()) {
            int u = que.front();
            que.pop();
            for (int j : g[u]) {
                int v = e[j].to;
                cap_t c = e[j].cap;
                if (c > 0 && h[v] == -1) {
                    h[v] = h[u] + 1;
                    if (v == t) return true;
                    que.push(v);
                }
            }
        }
        return false;
    }
    cap_t dfs(int u, int t, cap_t f) {
        if (u == t) return f;
        cap_t r = f;
        for (int &i = cur[u]; i < int(g[u].size()); ++i) {
            int j = g[u][i];
            int v = e[j].to;
            cap_t c = e[j].cap;
            if (c > 0 && h[v] == h[u] + 1) {
                cap_t a = dfs(v, t, min(r, c));
                e[j].cap -= a;
                e[j ^ 1].cap += a;
                r -= a;
                if (r == 0) return f;
            }
        }
        return f - r;
    }
};

```

```

    }
    void addEdge(int u, int v, cap_t c) {
        g[u].push_back(e.size());
        e.emplace_back(v, c);
        g[v].push_back(e.size());
        e.emplace_back(u, 0);
    }
    cap_t maxFlow(int s, int t) {
        cap_t ans = 0;
        while (bfs(s, t)) {
            cur.assign(n, 0);
            ans += dfs(s, t, INF);
        }
        return ans;
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m, source, sink;
    cin >> n >> m >> source >> sink;
    source--, sink--;
    Flow<ll> flow(n);
    for (int i = 0; i < m; ++i) {
        int u, v, c;
        cin >> u >> v >> c;
        u--, v--;
        flow.addEdge(u, v, c);
    }

    cout << flow.maxFlow(source, sink) << "\n";

    return 0;
}

```

// test problem: <https://loj.ac/p/101>

3.8 spfa.cpp

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

```
using namespace std;
using ll = long long;
```

```
const int inf = 1e9;
```

```

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

    vector<vector<pair<int, int>>> g(n);
    vector<int> w(m);
    for (int i = 0; i < m; ++i) {
        int u, v;
        cin >> u >> v >> w[i];
        u--, v--;
        g[u].emplace_back(v, i);
        if (w[i] >= 0) {
            g[v].emplace_back(u, i);
        }
    }
}

```

```

auto spfa = [&](int s) { // true: no negative ring
    vector<int> dis(n, inf), cnt(n);
    vector<bool> vis(n);

```

```

    dis[s] = 0;
    vis[s] = true;
    queue<int> q;
    q.push(s);

    while (!q.empty()) {
        int u = q.front();
        q.pop();
        vis[u] = false;
        for (auto [v, j] : g[u]) {
            if (dis[v] > dis[u] + w[j]) {
                dis[v] = dis[u] + w[j];
                cnt[v] = cnt[u] + 1;
                if (cnt[v] >= n) {
                    return false;
                }
            }
            if (vis[v] == false) {
                q.push(v);
                vis[v] = true;
            }
        }
    }

    return true;
};

cout << (spfa(0) ? "NO\n" : "YES\n");
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int t;
    cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}

```

// test problem: <https://www.luogu.com.cn/problem/P3385>

3.9 匈牙利.cpp

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 505;
int n1, n2, m, match[maxn];
vector<int> g[maxn];
bool vis[maxn];
bool find(int u) {
    for (auto v : g[u]) {
        if (vis[v]) continue;
        vis[v] = 1;
        if (match[v] == 0 || find(match[v])) {
            match[v] = u;
            return 1;
        }
    }
    return 0;
}

int main() {

```

```

scanf("%d%d", &n1, &n2, &m);
while (m--) {
    int u, v;
    scanf("%d", &u, &v);
    g[u].push_back(v);
}
int ans = 0;
for (int i = 1; i <= n1; ++i) {
    memset(vis, false, sizeof(vis));
    if (find(i)) ++ans;
}
printf("%d\n", ans);
return 0;
}

```

4 Math

4.1 China.cpp

```

#include <bits/stdc++.h>
using namespace std;
#define IO ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
typedef long long ll;
using namespace std;
/**
 *gcd(a,mod)=d ;x,y,d=ax+by
 *extended_euclid(a,mod)=ax+by
 */
ll extended_euclid(ll a, ll mod, ll &x, ll &y)
{
    int d;
    if (mod == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    d = extended_euclid(mod, a % mod, y, x);
    y = y - a / mod * x;
    return d;
}
/**
 *x=mod[i](modw[i]) 0<i<len
 *prime[i]>0
 */
ll chinese_remainder(int mod[], int prime[], int len)
{
    ll res, i, d, x, y, n, m;
    res = 0;
    n = 1;
    for (i = 0; i < len; i++)
        n *= prime[i];
    for (i = 0; i < len; i++)
    {
        m = n / prime[i];
        extended_euclid(prime[i], m, x, y);
        res = (res + y * m * mod[i]) % n;
    }
    return (n + res % n) % n;
}

int main()
{
    int len, mod[12], prime[12];
    while (cin >> len)
    {
        for (int i = 0; i < len; i++)

```

```

        cin >> prime[i] >> mod[i];
        cout << chinese_remainder(mod, prime, len) << endl;
    }
}

```

4.2 Euler.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

//
int euler_phi(int n) {
    int ans = n;
    for (int i = 2; i * i <= n; i++)
        if (n % i == 0) {
            ans = ans / i * (i - 1);
            while (n % i == 0) n /= i;
        }
    if (n > 1) ans = ans / n * (n - 1);
    return ans;
}

vector<int> phi_table(int n) {
    vector<int> phi(n + 1);
    phi[1] = 1;
    for (int i = 2; i <= n; i++) {
        if (phi[i]) continue;
        for (int j = i; j <= n; j += i) {
            if (!phi[j]) phi[j] = j;
            phi[j] = phi[j] / i * (i - 1);
        }
    }
    return phi;
}

```

4.3 FFT.cpp

```

#include <bits/stdc++.h>
using namespace std;
#define PI acos(-1.0)
const int maxn = 5e5 + 5;
const int INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;
struct Complex {
    double r, i;
    Complex(double _r = 0.0, double _i = 0.0) { r = _r, i = _i; }
    Complex operator+(const Complex &b) { return Complex(r + b.r, i + b.i); }
    Complex operator-(const Complex &b) { return Complex(r - b.r, i - b.i); }
    Complex operator*(const Complex &b) { return Complex(r * b.r - i * b.i, r * b.i + i * b.r); }
};
/*
 *      FFTIFFT
 *      i      i
 *      len2
 */
/*
FFT
len2^k
on ==1 DFTon ==-1IDFT
*/
int rev[maxn];
void FFT(Complex y[], int len, int on) {
    int bit = 0;
    while ((1 << bit) < len)

```

```

    bit++;
    for (int i = 0; i <= len - 1; i++) { // y ,
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
        if (i < rev[i]) swap(y[i], y[rev[i]]); // if
    }
    for (int h = 2; h <= len; h <= 1) { // h
        Complex wn(cos(-on * 2 * PI / h), sin(-on * 2 * PI / h)); //
        for (int j = 0; j < len; j += h) { //
            Complex w(1, 0);
            for (int k = j; k < j + h / 2; k++) { //
                Complex u = y[k];
                Complex t = w * y[k + h / 2];
                y[k] = u + t; //
                y[k + h / 2] = u - t; //
                w = w * wn;
            }
        }
    }
    if (on == -1)
        for (int i = 0; i < len; i++)
            y[i].r /= len;
}

char s1[maxn], s2[maxn];
int ans[maxn];
Complex a[maxn], b[maxn];
int main() {
    int i, len1, len2, len;
    while (~scanf("%s%s", s1, s2)) {
        len1 = strlen(s1);
        len2 = strlen(s2);
        len = 1;
        while (len < (len1 << 1) || len < (len2 << 1))
            len <<= 1;
        for (i = 0; i < len1; i++)
            a[i] = Complex(s1[len1 - i - 1] - '0', 0);
        for (; i < len; i++)
            a[i] = Complex(0, 0);
        for (i = 0; i < len2; i++)
            b[i] = Complex(s2[len2 - i - 1] - '0', 0);
        for (; i < len; i++)
            b[i] = Complex(0, 0);
        FFT(a, len, 1);
        FFT(b, len, 1);
        for (i = 0; i < len; i++)
            a[i] = a[i] * b[i];
        FFT(a, len, -1);
        for (i = 0; i < len; i++)
            ans[i] = (int)(a[i].r + 0.5);
        len = len1 + len2 - 1;
        for (i = 0; i < len; i++) {
            ans[i + 1] += ans[i] / 10;
            ans[i] %= 10;
        }
        for (i = len; ans[i] <= 0 && i > 0; i--)
            ;
        for (; i >= 0; i--)
            printf("%d", ans[i]);
        putchar('\n');
    }
}

```

4.4 Lagrange.cpp

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

```

```

const ll mod = 1e9 + 7;
const int maxn = 1e6 + 10;
ll t, n, m, l, r;
ll a[maxn], sum[maxn], pre[maxn], suf[maxn], fac[maxn];
ll ksm(ll x, ll n, ll p) // xn%p
{
    ll res = 1;
    while (n) {
        if (n & 1)
            res = (res * x) % p;
        x = (x * x) % p;
        n >>= 1;
    }
    return res;
}
/* cal 0 n n+1
   a yi
   n
   x
*/
ll cal(ll x, ll *a, ll n) {
    if (x <= n)
        return a[x];
    ll ans = 0;
    pre[0] = x;
    suf[n + 1] = 1;
    for (int i = 1; i <= n; i++) //
        pre[i] = pre[i - 1] * (x - i) % mod;
    for (int i = n; i >= 0; i--) //
        suf[i] = suf[i + 1] * (x - i) % mod;
    for (int i = 0; i <= n; i++) { //
        ll f = fac[n - i] * fac[i] % mod; //
        if ((n - i) % 2 == 1)
            f *= -1; //
        if (i == 0)
            ans = (ans + a[i] * f % mod * 1LL * suf[i + 1] % mod) % mod; // Y
        else
            ans = (ans + a[i] * f % mod * pre[i - 1] % mod * suf[i + 1] % mod) % mod;
    }
    return (ans + mod) % mod; // mod
}

void init() {
    fac[0] = 1;
    for (int i = 1; i < maxn; i++) // N
        fac[i] = fac[i - 1] * i % mod;
    for (int i = 0; i < maxn; i++) //
        fac[i] = ksm(fac[i], mod - 2, mod);
}

int main() {
    init();
    ll n, k;
    scanf("%lld %lld", &n, &k);
    ll sum = 0;
    a[0] = 0;
    for (int i = 1; i <= k + 2; i++) {
        sum = (sum + ksm(i, k, mod)) % mod;
        a[i] = sum;
    }
    printf("%lld\n", cal(n, a, k + 1));
}

```

4.5 Lucas.cpp

```

#include <bits/stdc++.h>

```

```

using namespace std;
using ll = long long;

int P = 1e9 + 7;
// assume -P <= x < P
int norm(int x) {
    if (x < 0) x += P;
    if (x >= P) x -= P;
    return x;
}

template<class T>
T power(T a, ll b) {
    T res = 1;
    for (; b; b /= 2, a *= a) {
        if (b % 2) res *= a;
    }
    return res;
}

struct Z {
    int x;
    Z(int x = 0) : x(norm(x)) {}
    Z(int64_t x) : x(x % P) {}
    int val() const {
        return x;
    }
    Z operator-() const {
        return Z(norm(P - x));
    }
    Z inv() const {
        assert(x != 0);
        return power(*this, P - 2);
    }
    Z &operator*=(const Z &rhs) {
        x = int64_t(x) * rhs.x % P;
        return *this;
    }
    Z &operator+=(const Z &rhs) {
        x = norm(x + rhs.x);
        return *this;
    }
    Z &operator-=(const Z &rhs) {
        x = norm(x - rhs.x);
        return *this;
    }
    Z &operator/=(const Z &rhs) {
        return *this *= rhs.inv();
    }
    friend Z operator*(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res *= rhs;
        return res;
    }
    friend Z operator+(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res += rhs;
        return res;
    }
    friend Z operator-(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res -= rhs;
        return res;
    }
    friend Z operator/(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res /= rhs;
        return res;
    }
};

friend istream &operator>>(istream &is, Z &a) {
    int64_t v;
    is >> v;
    a = Z(v);
    return is;
}

friend ostream &operator<<(ostream &os, const Z &a) {
    return os << a.val();
}

};

struct Binom {
    const int N;
    vector<Z> fac, invfac;
    Binom(int n) : N(n), fac(N + 1), invfac(N + 1) {
        fac[0] = 1;
        for (int i = 1; i <= N; i++) {
            fac[i] = fac[i - 1] * i;
        }
        invfac[N] = fac[N].inv();
        for (int i = N; i; i--) {
            invfac[i - 1] = invfac[i] * i;
        }
    }

    Z get(int n, int m) {
        if (m < 0 || n < m) return Z(0);
        return fac[n] * invfac[m] * invfac[n - m];
    }
};

void solve() {
    int n, m;
    cin >> n >> m >> P;

    Binom binom(P - 1);

    function<ll(int, int, int)> Lucas = [&](int n, int m, int P) {
        if (m == 0) return 1LL;
        return 1LL * binom.get(n % P, m % P).val() * Lucas(n / P, m / P, P) % P;
    };

    cout << Lucas(n + m, m, P) << "\n";
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int t;
    cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}

// test problem: https://www.luogu.com.cn/problem/P3807

```

4.6 Miller-Rabin.cpp

```

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

uint64_t mod_mul64(uint64_t a, uint64_t b, uint64_t mod) {
    assert(a < mod && b < mod);

```

```

if (mod <= 1LLU << 32)
    return a * b % mod;

if (mod <= 1LLU << 63) {
    uint64_t q = uint64_t((long double) a * b / mod);
    uint64_t result = a * b - q * mod;

    if (result > 1LLU << 63) {
        result += mod;
    } else if (result >= mod) {
        result -= mod;
    }

    return result;
}

#ifdef __SIZEOF_INT128__
    return uint64_t((__uint128_t(a) * b % mod);
#endif

assert(false);
}

uint64_t mod_pow64(uint64_t a, uint64_t b, uint64_t mod) {
    uint64_t result = 1;
    while (b > 0) {
        if (b & 1) {
            result = mod_mul64(result, a, mod);
        }
        a = mod_mul64(a, a, mod);
        b >>= 1;
    }
    return result;
}

bool miller_rabin(uint64_t n) {
    if (n < 2)
        return false;

    // Check small primes.
    for (uint64_t p : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29})
        if (n % p == 0)
            return n == p;

    // https://miller-rabin.appspot.com/
    auto get_miller_rabin_bases = [&]() -> vector<uint64_t> {
        if (n < 341531) return {9345883071009581737LLU};
        if (n < 1050535501) return {336781006125, 9639812373923155};
        if (n < 350269456337) return {4230279247111683200, 14694767155120705706LLU,
            16641139526367750375LLU};
        if (n < 55245642489451) return {2, 141889084524735, 1199124725622454117,
            11096072698276303650LLU};
        if (n < 7999252175582851) return {2, 4130806001517, 149795463772692060,
            186635894390467037, 3967304179347715805};
        if (n < 585226005592931977) return {2, 123635709730000, 9233062284813009,
            43835965440333360, 761179012939631437, 1263739024124850375};
        return {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
    };

    int r = __builtin_ctzll(n - 1);
    uint64_t d = (n - 1) >> r;

    for (uint64_t a : get_miller_rabin_bases()) {
        if (a % n == 0)
            continue;

```

```

uint64_t x = mod_pow64(a % n, d, n);

if (x == 1 || x == n - 1)
    continue;

for (int i = 0; i < r - 1 && x != n - 1; i++)
    x = mod_mul64(x, x, n);

if (x != n - 1)
    return false;
}

return true;
}

// Solution to https://www.spoj.com/problems/PON/
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int t;
    cin >> t;

    while (t--) {
        uint64_t n;
        cin >> n;
        cout << (miller_rabin(n) ? "YES" : "NO") << '\n';
    }

    return 0;
}

```

4.7 NTT.cpp

```

#include <bits/stdc++.h>
using namespace std;
#define ll long long
const int maxn = 2e5 + 10;
const ll mod = 998244353, g = 3;
int rev[maxn];
ll ksm(ll x, ll n, ll mod) { // x^n % mod
    ll res = 1;
    while (n) {
        if (n & 1) res = (res * x) % mod;
        x = (x * x) % mod;
        n >>= 1;
    }
    return res;
}

void NTT(ll y[], int len, int on) {
    int bit = 0;
    while ((1 << bit) < len) bit++;
    for (int i = 0; i <= len - 1; i++) // y
    {
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
        if (i < rev[i])
            swap(y[i], y[rev[i]]); // if
    }

    // FFT
    for (int h = 2; h <= len; h <= 1) // h
    {
        ll wn = ksm(g, (mod - 1) / h, mod); //
        if (on == -1) wn = ksm(wn, mod - 2, mod); //
        for (int j = 0; j < len; j += h) //
        {
            ll w = 1;
            for (int k = j; k < j + h / 2; k++) //

```

```

    {
        ll u = y[k];
        ll t = (w * y[k + h / 2]) % mod;
        y[k] = (u + t) % mod;
        y[k + h / 2] = (u - t + mod) % mod;
        w = (w * wn) % mod;
    }
}
if (on == -1) {
    ll t = ksm(len, mod - 2, mod);
    for (int i = 0; i < len; i++) y[i] = (y[i] * t) % mod;
}
char st[maxn], st1[maxn];
ll A[maxn], B[maxn];
int n;
int main() {
    while (~scanf("%s %s", st, st1)) {
        int len = strlen(st), len1 = strlen(st1);
        n = 1;
        while (n < (len << 1) || n < (len1 << 1)) n <<= 1;
        for (int i = 0; i < len; i++) A[len - 1 - i] = st[i] - '0';
        for (int i = len; i <= n; i++) A[i] = 0;
        for (int i = 0; i < len1; i++) B[len1 - 1 - i] = st1[i] - '0';
        for (int i = len1; i <= n; i++) B[i] = 0;
        NTT(A, n, 1);
        NTT(B, n, 1);
        for (int i = 0; i <= n - 1; i++) A[i] = A[i] * B[i] % mod;
        NTT(A, n, -1);
        for (int i = 0; i <= n - 1; i++) {
            A[i + 1] += A[i] / 10;
            A[i] %= 10;
        }
        n--;
        while (A[n] / 10) A[n + 1] += A[n] / 10, A[n++] %= 10;
        while (!A[n] && n > 0) n--;
        for (int i = n; i >= 0; i--) printf("%lld", A[i]);
        printf("\n");
    }
    return 0;
}

```

4.8 basic.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

template <typename T>
T floor(T a, T n) {
    if (n < 0) {
        n = -n;
        a = -a;
    }
    return a < 0 ? (a - n + 1) / n : a / n;
}

template <typename T>
T ceil(T a, T n) {
    if (n < 0) {
        n = -n;
        a = -a;
    }
    return a < 0 ? a / n : (a + n - 1) / n;
}

```

4.9 binom.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

constexpr int mod = 1e9 + 7;
// assume -mod <= x < 2mod
int norm(int x) {
    if (x < 0) x += mod;
    if (x >= mod) x -= mod;
    return x;
}

template<class T>
T power(T a, ll b) {
    T res = 1;
    for (; b; b /= 2, a *= a) {
        if (b % 2) res *= a;
    }
    return res;
}

struct Z {
    int x;
    Z(int x = 0) : x(norm(x)) {}
    Z(int64_t x) : x(x % mod) {}
    int val() const {
        return x;
    }
    Z operator-() const {
        return Z(norm(mod - x));
    }
    Z inv() const {
        assert(x != 0);
        return power(*this, mod - 2);
    }
    Z &operator*=(const Z &rhs) {
        x = int64_t(x) * rhs.x % mod;
        return *this;
    }
    Z &operator+=(const Z &rhs) {
        x = norm(x + rhs.x);
        return *this;
    }
    Z &operator-=(const Z &rhs) {
        x = norm(x - rhs.x);
        return *this;
    }
    Z &operator/=(const Z &rhs) {
        return *this *= rhs.inv();
    }
    friend Z operator*(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res *= rhs;
        return res;
    }
    friend Z operator+(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res += rhs;
        return res;
    }
    friend Z operator-(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res -= rhs;
        return res;
    }
    friend Z operator/(const Z &lhs, const Z &rhs) {

```

```

        Z res = lhs;
        res /= rhs;
        return res;
    }
    friend istream &operator>>(istream &is, Z &a) {
        int64_t v;
        is >> v;
        a = Z(v);
        return is;
    }
    friend ostream &operator<<(ostream &os, const Z &a) {
        return os << a.val();
    }
}

struct Binom {
    const int N;
    vector<Z> fac, invfac;
    Binom(int n) : N(n), fac(N + 1), invfac(N + 1) {
        fac[0] = 1;
        for (int i = 1; i <= N; i++) {
            fac[i] = fac[i - 1] * i;
        }
        invfac[N] = fac[N].inv();
        for (int i = N; i; i--) {
            invfac[i - 1] = invfac[i] * i;
        }
    }

    Z get(int n, int m) {
        if (m < 0 || n < m) return Z(0);
        return fac[n] * invfac[m] * invfac[n - m];
    }
};

```

4.10 exgcd.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

void solve() {
    ll a, b, c;
    cin >> a >> b >> c;

    // ax + by = gcd(a, b)
    // return tuple(d, x, y)
    function<tuple<int64_t, int64_t, int64_t>(int64_t, int64_t)> exgcd = [&](int64_t
        a, int64_t b) {
        if (b == 0) {
            return tuple(a, (int64_t)1, (int64_t)0);
        }
        auto [d, x, y] = exgcd(b, a % b);
        return tuple(d, y, x - a / b * y);
    };

    auto [d, x, y] = exgcd(a, b);

    if (c % d != 0) {
        cout << "-1\n";
    } else {
        x *= c / d;
        y *= c / d;

        ll dx = b / d;
        ll dy = a / d;
    }
}

```

```

        ll l = ceil(1.0 * (-x + 1) / dx);
        ll r = floor(1.0 * (y - 1) / dy);

        if (l > r) {
            cout << x + l * dx << " " << y - r * dy << "\n";
        } else {
            ll minx = x + l * dx, maxx = x + r * dx;
            ll miny = y - r * dy, maxy = y - l * dy;
            cout << r - l + 1 << " " << minx << " " << miny << " " << maxx << " " <<
                maxy << "\n";
        }
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int t;
    cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}

```

// test problem: <https://www.luogu.com.cn/problem/P5656>

4.11 xor_basis.cpp

```

template<typename T, int BITS = 30>
struct xor_basis {
    // A list of basis values sorted in decreasing order, where each value has a
    // unique highest bit.
    vector<T> basis(BITS);
    int n = 0;

    T min_value(T start) const {
        if (n == BITS) {
            return 0;
        }
        for (int i = 0; i < n; i++) {
            start = min(start, start ^ basis[i]);
        }
        return start;
    }

    T max_value(T start = 0) const {
        if (n == BITS) {
            return (T(1) << BITS) - 1;
        }
        for (int i = 0; i < n; i++) {
            start = max(start, start ^ basis[i]);
        }
        return start;
    }

    bool add(T x) {
        x = min_value(x);
        if (x == 0) {
            return false;
        }

        basis[n++] = x;
    }
}

```



```

        cout << ans << endl;
    } else {
        ll x;
        cin >> x;
        x ^= ans;
        insert(++n, x);
    }
}
}
}

```

4.14 取模gauss.cpp

```

ll a[55][55], x[55];
ll lcm(ll a, ll b) {
    return a / __gcd(a, b) * b;
}
ll pow2(ll a, ll b) {
    ll res = 1;
    while (b) {
        if (b & 1) res = res * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return res;
}
ll inv(ll a, ll m) {
    return pow2(a, mod - 2);
}
ll Gauss(ll m, ll n) {
    ll r = 0, c = 0;
    while (r < m && c < n) {
        ll id = r;
        for (ll i = r + 1; i < m; ++i)
            if (abs(a[i][c]) > abs(a[id][c]))
                id = i;
        if (id != r)
            for (ll i = 0; i <= n; ++i)
                swap(a[r][i], a[id][i]);
        if (abs(a[r][c]) != 0) {
            for (ll i = r + 1; i < m; ++i) {
                if (abs(a[i][c]) == 0) continue;
                ll LCM = lcm(abs(a[i][c]), abs(a[r][c]));
                ll ta = LCM / abs(a[i][c]);
                ll tb = LCM / abs(a[r][c]);
                if (a[i][c] * a[r][c] < 0) tb = -tb;
                for (ll j = c; j <= n; ++j)
                    a[i][j] = ((a[i][j] * ta - a[r][j] * tb) % mod + mod) % mod;
            }
            ++r;
        }
        ++c;
    }
    for (ll i = r; i < m; ++i)
        if (a[i][n] != 0) return -1;
    if (r < n) return n - r; // m
    for (ll i = n - 1; i >= 0; --i) {
        ll tmp = a[i][n];
        for (ll j = i + 1; j < n; ++j) {
            if (a[i][j] != 0) {
                tmp -= a[i][j] * x[j];
                tmp = (tmp % mod + mod) % mod;
            }
        }
        x[i] = (tmp * inv(a[i][i], mod)) % mod;
        debug(i, x[i])
    }
}

```

```

    return 0;
}

```

4.15 容斥.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

constexpr int mod = 998244353;
// assume -mod <= x < 2mod
int norm(int x) {
    if (x < 0) x += mod;
    if (x >= mod) x -= mod;
    return x;
}

template<class T>
T power(T a, int b) {
    T res = 1;
    for (; b; b /= 2, a *= a)
        if (b % 2) res *= a;
    return res;
}

struct Z {
    int x;
    Z(int x = 0) : x(norm(x)) {}
    Z(ll x) : x(x % mod) {}

    int val() const {
        return x;
    }
    Z operator-(const Z &rhs) const {
        return Z(norm(mod - x));
    }
    Z inv() const {
        assert(x != 0);
        return power(*this, mod - 2);
    }
    Z &operator*=(const Z &rhs) {
        x = ll(x) * rhs.x % mod;
        return *this;
    }
    Z &operator+=(const Z &rhs) {
        x = norm(x + rhs.x);
        return *this;
    }
    Z &operator-=(const Z &rhs) {
        x = norm(x - rhs.x);
        return *this;
    }
    Z &operator/=(const Z &rhs) {
        return *this *= rhs.inv();
    }
    friend Z operator*(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res *= rhs;
        return res;
    }
    friend Z operator+(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res += rhs;
        return res;
    }
    friend Z operator-(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res -= rhs;
    }
}

```

```

        return res;
    }
    friend Z operator/(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res /= rhs;
        return res;
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, L;
    cin >> n >> L;
    vector<int> s(n);
    for (int i = 0; i < n; ++i) {
        string t;
        cin >> t;
        for (auto c : t) {
            s[i] |= 1 << (c - 'a');
        }
    }

    auto cul = [&](int cur) {
        int ans = 0;
        while (cur) {
            ans += cur & 1;
            cur >>= 1;
        }
        return ans;
    };

    Z ans = 0;
    vector<Z> f(1 << n);
    for (int mask = 1; mask < (1 << n); ++mask) {
        int cur = (1 << 26) - 1;
        for (int i = 0; i < n; ++i) {
            if (mask >> i & 1) {
                cur &= s[i];
            }
        }
        f[mask] = power(Z(cul(cur)), L);
        ans += (cul(mask) & 1 ? 1 : -1) * f[mask];
    }

    cout << ans.val() << "\n";

    return 0;
}

```

// test problem: https://atcoder.jp/contests/abc246/tasks/abc246_f

4.16 异或gauss.cpp

```

#include <math.h>
#include <stdio.h>
#include <string.h>

#include <algorithm>
#include <iostream>
using namespace std;

const int MAXN = 50;

int a[MAXN][MAXN]; //
int x[MAXN]; //

```

```

int free_x[MAXN]; //

//      (Gauss-Jordan elimination)      .(-2
//      -100)
//      equuarequ ,0equ -1,var +1,0var.
int Gauss(int equ, int var) {
    int i, j, k;
    int max_r; //
    int col; //

    for (int i = 0; i <= var; i++) {
        x[i] = 0;
        free_x[i] = 1;
    }

    //
    col = 0;
    for (k = 0; k < equ && col < var; k++, col++) { //
        //      colk      .()
        max_r = k;
        for (i = k + 1; i < equ; i++) {
            if (abs(a[i][col]) > abs(a[max_r][col]))
                max_r = i;
        }
        if (max_r != k) { //      k.
            for (j = k; j < var + 1; j++)
                swap(a[k][j], a[max_r][j]);
        }
        if (a[k][col] == 0) { //      colk0.
            k--;
            continue;
        }
        for (i = k + 1; i < equ; i++) { //
            if (a[i][col] != 0) {
                for (j = col; j < var + 1; j++)
                    a[i][j] ^= a[k][j];
            }
        }
        // 1. :      (0, 0, ..., a) (a != 0).
        for (i = k; i < equ; i++) { //
            if (a[i][col] != 0)
                return -1;
        }
        return var - k;
    }

    int start[MAXN];
    int en[MAXN];

    int main() {
        // freopen("in.txt", "r", stdin);
        // freopen("out.txt", "w", stdout);
        int u, v;
        int T;
        int n;
        scanf("%d", &T);
        while (T--) {
            scanf("%d", &n);
            for (int i = 0; i < n; i++)
                scanf("%d", &start[i]);
            for (int i = 0; i < n; i++)
                scanf("%d", &en[i]);
            memset(a, 0, sizeof(a));
            while (scanf("%d%d", &u, &v)) {
                if (u == 0 && v == 0)
                    break;
            }
        }
    }
}

```

```

        a[v - 1][u - 1] = 1;
    }
    for (int i = 0; i < n; i++)
        a[i][i] = 1;
    for (int i = 0; i < n; i++)
        a[i][n] = start[i] ^ en[i];
    int ans = Gauss(n, n);
    if (ans == -1)
        printf("Oh,it's impossible-!!\n");
    else
        printf("%d\n", 1 << ans);
}
return 0;
}

```

4.17 斐波那契.cpp

```

#include <bits/stdc++.h>
#define rep(i, a, n) for (int i = a; i <= n; ++i)
#define per(i, a, n) for (int i = n; i >= a; --i)
#ifdef LOCAL
#include "Print.h"
#define de(...) W(['', #__VA_ARGS__,] "=", __VA_ARGS__)
#else
#define de(...)
#endif
using namespace std;
typedef long long ll;
const int maxn = 2e5 + 5;
const ll mod = 1e9 + 9;
void add(ll &x, ll y) { if ((x += y) >= mod) x -= mod; }
void sub(ll &x, ll y) { if ((x -= y) < 0) x += mod; }
struct mat {
    ll a[3][3];
    mat(int op) {
        if (op == 1) a[1][1] = a[2][2] = 1, a[1][2] = a[2][1] = 0;
        if (op == 0) a[1][1] = a[1][2] = a[2][1] = a[2][2] = 0;
    }
    mat operator*(const mat &A) {
        mat ans(0);
        rep(i, 1, 2) rep(j, 1, 2) rep(k, 1, 2)
            add(ans.a[i][j], a[i][k] * A.a[k][j] % mod);
        return ans;
    }
};
mat powmod(mat a, ll b) {
    mat ans(1);
    while (b) {
        if (b & 1) ans = ans * a;
        b >>= 1; a = a * a;
    }
    return ans;
}
ll powmod(ll a, ll b) {
    ll ans = 1;
    while (b) {
        if (b & 1) ans = ans * a % mod;
        b >>= 1; a = a * a % mod;
    }
    return ans;
}
int case_Test() {
    auto f = [&](ll n) -> ll {
        if (n == 1) return 1;
        if (n == 2) return 2;
        mat A(0);
        A.a[1][1] = A.a[1][2] = A.a[2][1] = 1;
    };
}

```

```

        A = powmod(A, n - 2);
        return (A.a[1][1] * 2 + A.a[1][2]) % mod;
    };
    ll n;
    scanf("%lld", &n);
    printf("%lld\n", f(n));
    return 0;
}
int main() {
#ifdef LOCAL
    freopen("/Users/chenjinglong/cpp_code/in.in", "r", stdin);
    freopen("/Users/chenjinglong/cpp_code/out.out", "w", stdout);
    clock_t start = clock();
#endif
    int _ = 1;
    scanf("%d", &_);
    while (_--) case_Test();
#ifdef LOCAL
    printf("Time used: %.3lfs\n", (double)(clock() - start) / CLOCKS_PER_SEC);
#endif
    return 0;
}

```

4.18 求逆元.cpp

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll Mod = 1e9 + 7;
ll exgcd(ll a, ll b, ll &x, ll &y) { //
    if (b == 0) {
        x = 1, y = 0;
        return a;
    }
    ll ret = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return ret;
}
ll getInv(int a, int mod) { //      amod-1s
    ll x, y;
    ll d = exgcd(a, mod, x, y);
    return d == 1 ? (x % mod + mod) % mod : -1;
}

int main() {
    ll x = getInv(24, Mod);
    int T;
    scanf("%d", &T);
    while (T--) {
        ll n;
        ll ans = 1;
        scanf("%lld", &n);
        for (ll i = n; i <= n + 3; ++i)
            ans = (ans * i) % Mod;
        ans = ans * x % Mod;
        printf("%lld\n", ans);
    }
}

```

4.19 浮点型gauss.cpp

```

#include <cmath>
#include <iostream>
using namespace std;
#define eps 1e-9
const int maxn = 5e2 + 5;
double a[maxn][maxn]; //

```

```

double x[maxn];          //
int n;
int gauss() {
    for (int i = 1; i <= n; i++) {          //
        int max_r = i;                      // i+1--na[j][i]
        for (int j = i + 1; j <= n; j++)    // i+1 n
            if (fabs(a[j][i]) > fabs(a[max_r][i]))
                max_r = j;

        for (int k = 1; k <= n + 1; k++)    // max_r i
            swap(a[max_r][k], a[i][k]);
        if (fabs(a[i][i]) < eps)
            continue;                      // a[i][i] > 0, 0,
        double p = a[i][i];                // a[i][i]
        for (int j = 1; j <= n + 1; j++)
            a[i][j] /= p;
        for (int j = i + 1; j <= n; j++)    // a[j][i]
        {
            if (i != j) {
                double tmp = a[j][i];
                for (int k = 1; k <= n + 1; k++)
                    a[j][k] -= a[i][k] * tmp;    // a[j][k] = a[j][k] - (a[i][k] / a[i][i]) * a[j][i];
            }
        }
    }
    int free_num = 0; //
    for (int i = 1; i <= n; i++) {
        int ans = 0;
        for (int j = 1; j <= n + 1; j++)    //
            if (fabs(a[i][j]) < eps) ans++;
        if (ans == n && a[i][n + 1])
            return -1;
        if (ans == n + 1)
            free_num++;
    }
    if (!free_num) { //
        for (int i = n - 1; i >= 1; i--)
            for (int j = i + 1; j <= n; j++)
                a[i][n + 1] -= a[j][n + 1] * a[i][j]; //
        for (int i = 1; i <= n; i++)
            x[i] = a[i][n + 1];
        return free_num;
    }
}

int main() {
    cin >> n;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n + 1; j++)
            cin >> a[i][j];
    int t = gauss();
    if (t == 0) {
        for (int i = 1; i <= n; i++) {
            if (fabs(x[i]) < eps)
                printf("0\n");
            else
                printf("%.2f\n", x[i]);
        }
    } else
        cout << "No Solution\n";
}

```

4.20 第二类斯特林数.cpp

```

#pragma region
#include <algorithm>

```

```

#include <cmath>
#include <cstring>
#include <iomanip>
#include <iostream>
#include <map>
#include <queue>
#include <set>
#include <stack>
#include <string>
#include <unordered_map>
#include <vector>
using namespace std;
typedef long long ll;
#define rep(i, a, n) for (int i = a; i <= n; ++i)
#define per(i, a, n) for (int i = n; i >= a; --i)
namespace fastIO {
#define BUF_SIZE 100000
#define OUT_SIZE 100000
//fread->R
bool IOError = 0;
//inline char nc(){char ch=getchar();if(ch==-1)IOError=1;return ch;}
inline char nc() {
    static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
    if (p1 == pend) {
        p1 = buf;
        pend = buf + fread(buf, 1, BUF_SIZE, stdin);
        if (pend == p1) {
            IOError = 1;
            return -1;
        }
    }
    return *p1++;
}
inline bool blank(char ch) { return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t'; }
}
template <class T>
inline bool R(T &x) {
    bool sign = 0;
    char ch = nc();
    x = 0;
    for (; blank(ch); ch = nc());
    if (IOError) return false;
    if (ch == '-') sign = 1, ch = nc();
    for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
    if (sign) x = -x;
    return true;
}
inline bool R(double &x) {
    bool sign = 0;
    char ch = nc();
    x = 0;
    for (; blank(ch); ch = nc());
    if (IOError) return false;
    if (ch == '-') sign = 1, ch = nc();
    for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
    if (ch == '.') {
        double tmp = 1;
        ch = nc();
        for (; ch >= '0' && ch <= '9'; ch = nc())
            tmp /= 10.0, x += tmp * (ch - '0');
    }
    if (sign)
        x = -x;
    return true;
}
}

```

```

inline bool R(char *s) {
    char ch = nc();
    for (; blank(ch); ch = nc())
        ;
    if (IOerror)
        return false;
    for (; !blank(ch) && !IOerror; ch = nc())
        *s++ = ch;
    *s = 0;
    return true;
}
inline bool R(char &c) {
    c = nc();
    if (IOerror) {
        c = -1;
        return false;
    }
    return true;
}
template <class T, class... U>
bool R(T &h, U &... t) { return R(h) && R(t...); }
#undef OUT_SIZE
#undef BUF_SIZE
}; // namespace fastIO
using namespace fastIO;
template <class T>
void _W(const T &x) { cout << x; }
void _W(const int &x) { printf("%d", x); }
void _W(const int64_t &x) { printf("%lld", x); }
void _W(const double &x) { printf("%.16f", x); }
void _W(const char &x) { putchar(x); }
void _W(const char *x) { printf("%s", x); }
template <class T, class U>
void _W(const pair<T, U> &x) { _W(x.F), putchar(' '), _W(x.S); }
template <class T>
void _W(const vector<T> &x) {
    for (auto i = x.begin(); i != x.end(); _W(*i++))
        if (i != x.cbegin()) putchar(' ');
}
void W() {}
template <class T, class... U>
void W(const T &head, const U &... tail) { _W(head), putchar(sizeof...(tail) ? ' ' : '\n'), W(tail...); }
#pragma endregion
const int maxn = 1005;
const ll mod = 1e9 + 7;

ll Stirling[maxn][maxn], fac[maxn];
void init() {
    fac[1] = 1;
    rep(i, 2, 1000) fac[i] = fac[i - 1] * i % mod;
    Stirling[0][0] = 0;
    Stirling[1][1] = 1;
    for (ll i = 2; i < maxn; i++)
        for (ll j = 1; j <= i; j++)
            Stirling[i][j] = (Stirling[i - 1][j - 1] + j * Stirling[i - 1][j]) % mod;
}

```

4.21 线性基类.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

struct L_B {
    ll b[61], p[61]; //

```

```

    int cnt, flag; // , 0
    L_B() {
        memset(b, 0, sizeof(b));
        memset(p, 0, sizeof(p));
        cnt = 0, flag = 0;
    }
    inline bool insert(ll x) {
        for (int i = 60; i >= 0 && x; i--)
            if (x & (1LL << i)) {
                if (b[i]) x ^= b[i];
                else {
                    b[i] = x;
                    return true;
                }
            }
        flag = 1;
        return false;
    }
    ll qmax() {
        ll ans = 0;
        for (int i = 60; i >= 0; i--)
            if ((ans ^ b[i]) > ans) ans ^= b[i];
        return ans;
    }
    ll qmin() {
        if (flag) return 0;
        for (int i = 0; i <= 60; i++)
            if (b[i]) return b[i];
        return 0;
    }
    inline void rebuild() {
        for (int i = 60; i >= 1; i--) {
            if (b[i])
                for (int j = i - 1; j >= 0; j--)
                    if (b[i] & (1LL << j)) b[i] ^= b[j];
        }
        // p[i] 1<=i
        for (int i = 0; i <= 60; i++)
            if (b[i]) p[cnt++] = b[i];
    }
    ll kth(ll k) {
        if (flag) --k;
        if (!k) return 0;
        ll ans = 0;
        if (k >= (1LL << cnt)) return -1;
        for (int i = 0; i <= cnt; ++i)
            if (k & (1LL << i)) ans ^= p[i];
        return ans;
    }
};
L_B merge(const L_B &n1, const L_B &n2) {
    L_B ans = n1;
    for (int i = 60; i >= 0; i--)
        if (n2.b[i]) ans.insert(n2.b[i]);
    ans.flag = n1.flag | n2.flag;
    return ans;
}

```

4.22 除法分块.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

int main() {
    ios::sync_with_stdio(false);

```

```

cin.tie(nullptr);

//  $n / l = n / (l + 1) = \dots = n / r, 1 \leq l \leq r \leq k$ 
auto block = [&](int n, int k) {
    vector<array<int, 2>> ans;
    for (int l = 1, r; l <= k; l = r + 1) {
        r = (n / l ? min(k, n / (n / l)) : k);
        ans.push_back({l, r});
    }
    for (auto [l, r] : ans) {
        cout << l << " " << r << " " << n / l << "\n";
    }
};

block(24, 24);

return 0;
}

```

5 Others

5.1 BigNum2.cpp

```

// #include <bits/stdc++.h>
#include <iostream>
#include <vector>
using namespace std;
struct BigNum : vector<int> //      vector
{
    //
    //
    BigNum(int n = 0) //      00
    {
        push_back(n);
        check();
    }
    BigNum &check() //
    {
        while (!empty() && !back())
            pop_back(); //      0
        if (empty())
            return *this;
        for (int i = 1; i < size(); ++i) //
        {
            (*this)[i] += (*this)[i - 1] / 10;
            (*this)[i - 1] %= 10;
        }
        while (back() >= 10) {
            push_back(back() / 10);
            (*this)[size() - 2] %= 10;
        }
        return *this; //
    }
};
//
istream &operator>>(istream &is, BigNum &n) {
    string s;
    is >> s;
    n.clear();
    for (int i = s.size() - 1; i >= 0; --i)
        n.push_back(s[i] - '0');
    return is;
}
ostream &operator<<(ostream &os, const BigNum &n) {
    if (n.empty())
        os << 0;
    for (int i = n.size() - 1; i >= 0; --i)

```

```

        os << n[i];
    return os;
}
//
//
bool operator!=(const BigNum &a, const BigNum &b) {
    if (a.size() != b.size())
        return 1;
    for (int i = a.size() - 1; i >= 0; --i)
        if (a[i] != b[i])
            return 1;
    return 0;
}
bool operator==(const BigNum &a, const BigNum &b) {
    return !(a != b);
}
bool operator<(const BigNum &a, const BigNum &b) {
    if (a.size() != b.size())
        return a.size() < b.size();
    for (int i = a.size() - 1; i >= 0; --i)
        if (a[i] != b[i])
            return a[i] < b[i];
    return 0;
}
bool operator>(const BigNum &a, const BigNum &b) {
    return b < a;
}
bool operator<=(const BigNum &a, const BigNum &b) {
    return !(a > b);
}
bool operator>=(const BigNum &a, const BigNum &b) {
    return !(a < b);
}
//      +=
BigNum &operator+=(BigNum &a, const BigNum &b) {
    if (a.size() < b.size())
        a.resize(b.size());
    for (int i = 0; i != b.size(); ++i)
        a[i] += b[i];
    return a.check();
}
BigNum operator+(BigNum a, const BigNum &b) {
    return a += b;
}
//
BigNum &operator--(BigNum &a, BigNum b) {
    if (a < b)
        swap(a, b);
    for (int i = 0; i != b.size(); a[i] -= b[i], ++i)
        if (a[i] < b[i]) //
        {
            int j = i + 1;
            while (!a[j])
                ++j;
            while (j > i) {
                --a[j];
                a[--j] += 10;
            }
        }
    return a.check();
}
BigNum operator-(BigNum a, const BigNum &b) {
    return a -= b;
}
//      *=
BigNum operator*(const BigNum &a, const BigNum &b) {
    BigNum n;

```

```

    n.assign(a.size() + b.size() - 1, 0);
    for (int i = 0; i != a.size(); ++i)
        for (int j = 0; j != b.size(); ++j)
            n[i + j] += a[i] * b[j];
    return n.check();
}
BigNum &operator*=(BigNum &a, const BigNum &b) {
    return a = a * b;
}
//
BigNum divmod(BigNum &a, const BigNum &b) {
    BigNum ans;
    for (int t = a.size() - b.size(); a >= b; --t) {
        BigNum d;
        d.assign(t + 1, 0);
        d.back() = 1;
        BigNum c = b * d;
        while (a >= c) {
            a -= c;
            ans += d;
        }
    }
    return ans;
}
BigNum operator/(BigNum a, const BigNum &b) {
    return divmod(a, b);
}
BigNum &operator/=(BigNum &a, const BigNum &b) {
    return a = a / b;
}
BigNum &operator%=(BigNum &a, const BigNum &b) {
    divmod(a, b);
    return a;
}
BigNum operator%(BigNum a, const BigNum &b) {
    return a %= b;
}
//
BigNum pow(const BigNum &n, const BigNum &k) {
    if (k.empty())
        return 1;
    if (k == 2)
        return n * n;
    if (k.back() % 2)
        return n * pow(n, k - 1);
    return pow(pow(n, k / 2), 2);
}

int main() {
}

```

5.2 Simulated_annealing.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

const double eps = 1e-8;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;
}

```

```

vector<tuple<int, int, int>> a(n);
for (int i = 0; i < n; ++i) {
    int x, y, z;
    cin >> x >> y >> z;
    a[i] = tuple(x, y, z);
}

auto solve = [&]() {
    double step = 10000, ans = 1e30;
    tuple<double, double, double> tp;
    int pos = 0;

    auto dis = [&](auto A, auto B) {
        auto [x1, y1, z1] = A;
        auto [x2, y2, z2] = B;
        return sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1) + (z2 - z1) * (z2 - z1));
    };

    while (step > eps) {
        for (int i = 0; i < n; ++i) { //
            if (dis(tp, a[pos]) < dis(tp, a[i])) {
                pos = i;
            }
            double mt = dis(tp, a[pos]);
            ans = min(ans, mt);
            auto [x, y, z] = tp;
            auto [px, py, pz] = a[pos];
            x += (px - x) / mt * step;
            y += (py - y) / mt * step;
            z += (pz - z) / mt * step;
            tp = tuple(x, y, z);

            step *= 0.98;
        }
        return ans;
    };

    cout << fixed << setprecision(8) << solve() << "\n";

    return 0;
}

```

// test problem: <https://vjudge.net/problem/Gym-101981D>

5.3 Z.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

constexpr int mod = 1e9 + 7;
// assume -mod <= x < 2mod
int norm(int x) {
    if (x < 0) x += mod;
    if (x >= mod) x -= mod;
    return x;
}

template<class T>
T power(T a, int64_t b) {
    T res = 1;
    for (; b; b /= 2, a *= a) {
        if (b % 2) res *= a;
    }
    return res;
}

```



```

}
struct Z {
    int x;
    Z(int x = 0) : x(norm(x)) {}
    Z(int64_t x) : x(x % mod) {}
    int val() const {
        return x;
    }
    Z operator-() const {
        return Z(norm(mod - x));
    }
    Z inv() const {
        assert(x != 0);
        return power(*this, mod - 2);
    }
    Z &operator*=(const Z &rhs) {
        x = int64_t(x) * rhs.x % mod;
        return *this;
    }
    Z &operator+=(const Z &rhs) {
        x = norm(x + rhs.x);
        return *this;
    }
    Z &operator-=(const Z &rhs) {
        x = norm(x - rhs.x);
        return *this;
    }
    Z &operator/=(const Z &rhs) {
        return *this *= rhs.inv();
    }
    friend Z operator*(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res *= rhs;
        return res;
    }
    friend Z operator+(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res += rhs;
        return res;
    }
    friend Z operator-(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res -= rhs;
        return res;
    }
    friend Z operator/(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res /= rhs;
        return res;
    }
    friend istream &operator>>(istream &is, Z &a) {
        int64_t v;
        is >> v;
        a = Z(v);
        return is;
    }
    friend ostream &operator<<(ostream &os, const Z &a) {
        return os << a.val();
    }
};

```

5.4 bignum.cpp

```

#include <cstring>
#include <iostream>
using namespace std;

```

```

class BigNum {
private:
    int a[1000];
    int len;

public:
    BigNum() {
        len = 1;
        memset(a, 0, sizeof(a));
    }
    BigNum(const int b);
    BigNum(char *s);
    BigNum(const BigNum &T);
    BigNum &operator=(const BigNum &n);

    friend istream &operator>>(istream &, BigNum &);
    friend ostream &operator<<(ostream &, BigNum &);

    BigNum operator+(const BigNum &T) const;
    BigNum operator-(const BigNum &T) const;
    BigNum operator*(const BigNum &T) const;
    BigNum operator/(const int &b) const;
    BigNum operator|(const BigNum &T) const;
    BigNum operator%(const BigNum &T) const;

    bool operator>(const BigNum &T) const;
    bool operator>(const int &t) const;
};

BigNum::BigNum(const int b) {
    len = 0;
    memset(a, 0, sizeof(a));
    int t = b;
    while (t) {
        int x = t % 10;
        a[len++] = x;
        t /= 10;
    }
}

BigNum::BigNum(char *s) {
    memset(a, 0, sizeof(a));
    int l = strlen(s);
    len = l;
    int cnt = 0;
    for (int i = l - 1; i >= 0; --i)
        a[cnt++] = s[i] - '0';
}

BigNum::BigNum(const BigNum &T) : len(T.len) {
    memset(a, 0, sizeof(a));
    for (int i = 0; i < len; ++i)
        a[i] = T.a[i];
}

BigNum &BigNum::operator=(const BigNum &n) {
    len = n.len;
    memset(a, 0, sizeof(a));
    for (int i = 0; i < len; ++i)
        a[i] = n.a[i];
    return *this;
}

istream &operator>>(istream &in, BigNum &b) {
    char ch[1000];
    in >> ch;
    int l = strlen(ch);
    int count = 0;
    for (int i = l - 1; i > 0; --i) {
        b.a[count++] = ch[i] - '0';
    }
}

```

```

    if (ch[0] == '-')
        b.a[count - 1] = 0 - b.a[count - 1];
    else
        b.a[count++] = ch[0] - '0';
    b.len = count;
    return in;
}

ostream &operator<<(ostream &out, BigNum &b) {
    for (int i = b.len - 1; i >= 0; --i)
        cout << b.a[i];
    return out;
}

BigNum BigNum::operator+(const BigNum &T) const {
    BigNum t(*this);
    int big;
    big = T.len > len ? T.len : len;
    for (int i = 0; i < big; ++i) {
        t.a[i] += T.a[i];
        if (t.a[i] >= 10) {
            t.a[i + 1]++;
            t.a[i] -= 10;
        }
    }
    if (t.a[big] != 0)
        t.len = big + 1;
    else
        t.len = big;
    return t;
}

BigNum BigNum::operator-(const BigNum &T) const {
    int big;
    bool flag;
    BigNum t1, t2;
    if (*this > T) {
        t1 = *this;
        t2 = T;
        flag = 0;
    } else {
        t1 = T;
        t2 = *this;
        flag = 1;
    }
    big = t1.len;
    for (int i = 0; i < big; ++i) {
        if (t1.a[i] < t2.a[i]) {
            int j = i + 1;
            while (t1.a[j] == 0)
                j++;
            t1.a[j--]--;
            while (j > i)
                t1.a[j--] += 9;
            t1.a[i] += 10 - t2.a[i];
        } else
            t1.a[i] -= t2.a[i];
    }
    t1.len = big;
    while (t1.a[t1.len - 1] == 0 && t1.len > 1) {
        t1.len--;
        big--;
    }
    if (flag)
        t1.a[big - 1] = 0 - t1.a[big - 1];
    return t1;
}

BigNum BigNum::operator*(const BigNum &T) const {
    BigNum ret;
    int up;

    int temp, temp1;
    int i, j;
    for (i = 0; i < len; ++i) {
        up = 0;
        for (j = 0; j < T.len; ++j) {
            temp = a[i] * T.a[j] + ret.a[i + j] + up;
            if (temp >= 10) {
                temp1 = temp % 10;
                up = temp / 10;
                ret.a[i + j] = temp1;
            } else {
                up = 0;
                ret.a[i + j] = temp;
            }
        }
        if (up != 0)
            ret.a[i + j] = up;
    }
    ret.len = i + j;
    while (ret.a[ret.len - 1] == 0 && ret.len > 1)
        ret.len--;
    return ret;
}

BigNum BigNum::operator/(const int &b) const {
    BigNum ret;
    int down = 0;
    for (int i = len - 1; i >= 0; --i) {
        ret.a[i] = (a[i] + down * 10) / b;
        down = a[i] + down * 10 - ret.a[i] * b;
    }
    ret.len = len;
    while (ret.a[ret.len - 1] == 0 && ret.len > 1)
        ret.len--;
    return ret;
}

BigNum BigNum::operator|(const BigNum &T) const {
    BigNum ans;
    BigNum a = *this, b = T;
    int len1 = len, len2 = T.len;
    int t = len1 - len2;
    BigNum x = 1;
    BigNum ten = 10;
    for (int i = 0; i < t; ++i) {
        b = b * ten;
        x = x * ten;
    }
    while (a > T || (!a > T) && !(T > a)) {
        while (a > b || (!a > b) && !(b > a)) {
            a = a - b;
            ans = ans + x;
        }
        b = b / 10;
        x = x / 10;
    }
    return ans;
}

BigNum BigNum::operator%(const BigNum &T) const {
    BigNum ans;
    BigNum a = *this, b = T;
    int len1 = len, len2 = T.len;
    int t = len1 - len2;
    BigNum x = 1;
    BigNum ten = 10;
    for (int i = 0; i < t; ++i) {
        b = b * ten;
        x = x * ten;
    }
}

```

```

while (a > T || (!(a > T) && !(T > a))) {
    while (a > b || (!(a > b) && !(b > a))) {
        a = a - b;
        ans = ans + x;
    }
    b = b / 10;
    x = x / 10;
}
return a;
}

bool BigNum::operator>(const BigNum &T) const {
    int ln;
    if (len > T.len)
        return true;
    else if (len < T.len)
        return false;

    ln = len - 1;
    while (a[ln] == T.a[ln] && ln >= 0)
        ln--;
    if (ln >= 0 && a[ln] > T.a[ln])
        return true;
    else
        return false;
}

bool BigNum::operator>(const int &t) const {
    BigNum b(t);
    return *this > b;
}

```

5.5 gen.py

```

from random import *

# make data randint(1, r)

n = randint(1, 100000)

print(n)

```

5.6 makestd.cpp

```

#include <bits/stdc++.h>

using namespace std;
using ll = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string s;
    while (getline(cin, s)) {
        cout << "\"";
        for (auto it : s) {
            if (it == '"' || it == '\\')
                cout << "\\\"";
            cout << it;
        }
        cout << "\",\n";
        cout << endl;
    }
    return 0;
}

```

5.7 pai.py

```

import os

stdName = "A"
bfName = "B"
dirName = "pai"

os.system("g++ -std=c++20 -Wall {0:}.cpp -o std".format(stdName))
os.system("g++ -std=c++20 -Wall {0:}.cpp -o bf".format(bfName))

os.system("mkdir {0:}".format(dirName))
os.system("mv std {0:}".format(dirName))
os.system("mv bf {0:}".format(dirName))

tc = 0
while True:
    os.system("python gen.py > ./{0:}/in.in".format(dirName))
    os.system("time ./{0:}/std < ./{0:}/in.in > ./{0:}/std.out".format(dirName))
    os.system("./{0:}/bf < ./{0:}/in.in > ./{0:}/bf.out".format(dirName))
    if os.system("diff ./{0:}/bf.out ./{0:}/std.out".format(dirName)):
        print("WA")
        exit(0)
    else:
        tc += 1
        print("AC #", tc)

```

5.8 sg函数.cpp

```

#include <algorithm>
#include <cstring>
#include <iostream>
using namespace std;
#define IO ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
typedef long long ll;
const int maxm = 1e4 + 5;
const int maxn = 105;
int k;
int f[maxn], sg[maxm], vis[maxm]; //
void dosg() {
    sg[0] = 0;
    memset(vis, -1, sizeof(vis));
    for (int i = 1; i < maxm; ++i) {
        for (int j = 1; f[j] <= i && j <= k; ++j)
            vis[sg[i - f[j]]] = i;
        int j = 0;
        while (vis[j] == i)
            ++j;
        sg[i] = j;
    }
}

int main() {
    while (scanf("%d", &k) && k) {
        for (int i = 1; i <= k; ++i)
            scanf("%d", &f[i]);
        sort(f + 1, f + 1 + k);
        dosg();
        int m;
        scanf("%d", &m);
        while (m--) {
            int n;
            scanf("%d", &n);
            int ans = 0;
            for (int i = 1; i <= n; ++i) {
                int x;
                scanf("%d", &x);
                ans ^= sg[x];
            }
        }
    }
}

```

```

    }
    printf("%c", ans ? 'W' : 'L');
    // cout << (ans ? "W" : "L");
}
printf("\n");
}
}

```

5.9 博弈.cpp

```

#include <cmath>
#include <cstring>
#include <iostream>
#define gold (sqrt(5.0) + 1) / 2
using namespace std;
typedef long long ll;

int sg[1005];
const int N = 30;
int f[N];
int s[1005];
void DoSg(int num) {
    int i, j;
    memset(sg, 0, sizeof(sg));
    for (i = 1; i <= num; ++i) {
        memset(s, 0, sizeof(s));
        for (j = 0; f[j] <= i && j < N; ++j) {
            s[sg[i - f[j]]] = 1;
        }
        for (j = 0; ++j) {
            if (!s[j]) {
                sg[i] = j;
                break;
            }
        }
    }
}

int main() {
    ios::sync_with_stdio(false);
    f[0] = 1;
    f[1] = 1;
    for (int i = 2; i <= 30; ++i) {
        f[i] = f[i - 1] + f[i - 2];
    }
    DoSg(1000);
    int n, m, k;
    while (cin >> n >> m >> k) {
        if (n == 0 && m == 0 && k == 0)
            break;
        if (sg[n] ^ sg[m] ^ sg[k])
            cout << "Fibo" << endl;
        else
            cout << "Nqcci" << endl;
    }
}

```

5.10 威佐夫博弈.cpp

```

#include <algorithm>
#include <cmath>
#include <iostream>
#define gold (sqrt(5.0) + 1) / 2
using namespace std;
typedef long long ll;

int main() {

```

```

    ios::sync_with_stdio(false);
    int a, b;
    while (cin >> a >> b) {
        int big = max(a, b);
        int small = min(a, b);
        double now = double(big - small) * gold;
        if ((int)now == small)
            cout << 0 << endl; //
        else
            cout << 1 << endl; //
    }
}

```

5.11 杜教BM.cpp

```

#include <bits/stdc++.h>
using namespace std;
#define rep(i, a, n) for (long long i = a; i < n; i++)
#define per(i, a, n) for (long long i = n - 1; i >= a; i--)
#define pb push_back
#define all(x) (x).begin(), (x).end()
#define SZ(x) ((long long)(x).size())
typedef vector<long long> VI;
typedef long long ll;
typedef pair<long long, long long> PII;
const ll mod = 1e9 + 7;
ll powmod(ll a, ll b) {
    ll res = 1;
    a %= mod;
    assert(b >= 0);
    for (; b; b >>= 1) {
        if (b & 1)
            res = res * a % mod;
        a = a * a % mod;
    }
    return res;
}
// head

namespace linear_seq {
const long long N = 10010;
ll res[N], base[N], _c[N], _md[N];

vector<long long> Md;
void mul(ll *a, ll *b, long long k) {
    rep(i, 0, k + k) _c[i] = 0;
    rep(i, 0, k) if (a[i]) rep(j, 0, k)
        _c[i + j] = (_c[i + j] + a[i] * b[j]) % mod;
    for (long long i = k + k - 1; i >= k; i--)
        if (_c[i])
            rep(j, 0, SZ(Md)) _c[i - k + Md[j]] = (_c[i - k + Md[j]] - _c[i] * _md[Md[j]]) % mod;
    rep(i, 0, k) a[i] = _c[i];
}

long long solve(ll n, VI a, VI b) { // a b b[n+1]=a[0]*b[n]+...
    // printf("%d\n", SZ(b));
    ll ans = 0, pnt = 0;
    long long k = SZ(a);
    assert(SZ(a) == SZ(b));
    rep(i, 0, k) _md[k - 1 - i] = -a[i];
    _md[k] = 1;
    Md.clear();
    rep(i, 0, k) if (_md[i] != 0) Md.push_back(i);
    rep(i, 0, k) res[i] = base[i] = 0;
    res[0] = 1;
    while ((1ll << pnt) <= n) pnt++;
    for (long long p = pnt; p >= 0; p--) {

```

```

mul(res, res, k);
if ((n >> p) & 1) {
    for (long long i = k - 1; i >= 0; i--)
        res[i + 1] = res[i];
    res[0] = 0;
    rep(j, 0, SZ(Md)) res[Md[j]] = (res[Md[j]] - res[k] * _md[Md[j]]) % mod;
}
}
rep(i, 0, k) ans = (ans + res[i] * b[i]) % mod;
if (ans < 0) ans += mod;
return ans;
}
}
VI BM(VI s) {
    VI C(1, 1), B(1, 1);
    long long L = 0, m = 1, b = 1;
    rep(n, 0, SZ(s)) {
        ll d = 0;
        rep(i, 0, L + 1) d = (d + (ll)C[i] * s[n - i]) % mod;
        if (d == 0)
            ++m;
        else if (2 * L <= n) {
            VI T = C;
            ll c = mod - d * powmod(b, mod - 2) % mod;
            while (SZ(C) < SZ(B) + m)
                C.pb(0);
            rep(i, 0, SZ(B)) C[i + m] = (C[i + m] + c * B[i]) % mod;
            L = n + 1 - L;
            B = T;
            b = d;
            m = 1;
        } else {
            ll c = mod - d * powmod(b, mod - 2) % mod;
            while (SZ(C) < SZ(B) + m) C.pb(0);
            rep(i, 0, SZ(B)) C[i + m] = (C[i + m] + c * B[i]) % mod;
            ++m;
        }
    }
    return C;
}
}
long long gao(VI a, ll n) {
    VI c = BM(a);
    c.erase(c.begin());
    rep(i, 0, SZ(c)) c[i] = (mod - c[i]) % mod;
    return solve(n, c, VI(a.begin(), a.begin() + SZ(c)));
}
}; // namespace linear_seq

```

```

int main() {
    int n;
    cin >> n;
    cout << linear_seq::gao(VI{0, 1, 5, 18, 58, 177, 522, 1503, 4252, 11869}, n - 1)
        << "\n";
}

```

5.12 欧拉函数.cpp

```

//
// (n)=n*(1-1/p1)*(1-1/p2)*(1-1/p3)*(1-1/p4)...(1-1/pn)    pin
//
long long eular(long long n)
{
    long long ans = n;
    for (int i = 2; i * i <= n; i++)
    {
        if (n % i == 0)
        {
            ans -= ans / i;    //
        }
    }
}

```

```

while (n % i == 0) //    in
    n /= i;
}
}
if (n > 1)
    ans -= ans / n; //
return ans;
}

```

6 String

6.1 AhoCorasick.cpp

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

```
using namespace std;
using ll = long long;
```

```

/** Modified from:
 * https://github.com/kth-competitive-programming/kactl/blob/master/content/strings/
 * AhoCorasick.h
 * Try to handle duplicated patterns beforehand, otherwise change 'end' to
 * vector; empty patterns are not allowed. Time: construction takes  $O(26N)$ ,
 * where  $N$  = sum of length of patterns. find(x) is  $O(N)$ , where  $N$  = length of x.
 * findAll is  $O(N+M)$  where  $M$  is number of occurrence of all pattern (up to  $N\sqrt{N}$ )
 */
struct AhoCorasick {
    enum { alpha = 26, first = 'a' }; // change this!
    struct Node {
        // back: failure link, points to longest suffix that is in the trie.
        // end: longest pattern that ends here, is -1 if no patten ends here.
        // nmatches: number of (patterns that is a suffix of current node)/(
        // duplicated patterns), depends on needs.
        // output: output link, points to the longest pattern that is a suffix of
        // current node
        int back, end = -1, nmatches = 0, output = -1;
        array<int, alpha> ch;
        Node(int v = -1) { fill(ch.begin(), ch.end(), v); }
    };
    vector<Node> N;
    int n;
    AhoCorasick() : N(1), n(0) {}
    void insert(string &s) {
        assert(!s.empty());
        int p = 0;
        for (char c : s) {
            if (N[p].ch[c - first] == -1) {
                N[p].ch[c - first] = N.size();
                N.emplace_back();
            }
            p = N[p].ch[c - first];
        }
        N[p].end = n++;
        N[p].nmatches++;
    }
    void build() {
        N[0].back = (int)N.size();
        N.emplace_back(0);
        queue<int> q;
        q.push(0);
        while (!q.empty()) {
            int p = q.front();
            q.pop();
            for (int i = 0; i < alpha; i++) {
                int pnx = N[N[p].back].ch[i];
                auto &nxt = N[N[p].ch[i]];
                if (N[p].ch[i] == -1) N[p].ch[i] = pnx;
            }
        }
    }
}

```

```

    else {
        nxt.back = pnx;
        // if prev is an end node, then set output to prev node,
        // otherwise set to output link of prev node
        nxt.output = N[pnx].end == -1 ? N[pnx].output : pnx;
        // if we don't want to distinguish info of patterns that is
        // a suffix of current node, we can add info to the ch
        // node like this: nxt.nmatches+=N[pnx].nmatches;
        q.push(N[p].ch[i]);
    }
}
}
// for each position, finds the longest pattern that ends here
vector<int> find(const string &text) {
    int len = text.length();
    vector<int> res(len);
    int p = 0;
    for (int i = 0; i < len; i++) {
        p = N[p].ch[text[i] - first];
        res[i] = N[p].end;
    }
    return res;
}
// for each position, finds the all that ends here
vector<vector<int>> find_all(const string &text) {
    int len = text.length();
    vector<vector<int>> res(len);
    int p = 0;
    for (int i = 0; i < len; i++) {
        p = N[p].ch[text[i] - first];
        res[i].push_back(N[p].end);
        for (int ind = N[p].output; ind != -1; ind = N[ind].output) {
            assert(N[ind].end != -1);
            res[i].push_back(N[ind].end);
        }
    }
    return res;
}
int find_cnt(const string &text) {
    int len = text.length();
    vector<int> num(n + 1, 0);
    int p = 0, ans = 0;
    for (int i = 0; i < len; i++) {
        p = N[p].ch[text[i] - first];
        if (N[p].end != -1) {
            if (!num[N[p].end]) {
                num[N[p].end]++;
                ans += N[p].nmatches;
            }
        }
        for (int ind = N[p].output; ind != -1; ind = N[ind].output) {
            if (!num[N[ind].end]) {
                num[N[ind].end]++;
                ans += N[ind].nmatches;
            }
        }
    }
    return ans;
}
pair<int, vector<int>> find_maxcnt(const string &text) {
    int len = text.length();
    vector<int> num(n + 1, 0);
    int p = 0, ans = 0;
    for (int i = 0; i < len; i++) {
        p = N[p].ch[text[i] - first];
        if (N[p].end != -1) {

```

```

            if (!num[N[p].end]) {
                num[N[p].end]++;
                ans = max(ans, N[p].nmatches);
            }
        }
        for (int ind = N[p].output; ind != -1; ind = N[ind].output) {
            if (!num[N[ind].end]) {
                num[N[ind].end]++;
                ans += N[ind].nmatches;
            }
        }
    }
    vector<int> idx;
    for (int i = 0; i < n; i++) {
        if (num[i] == ans) {
            idx.push_back(i);
        }
    }
    return pair(ans, idx);
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    AhoCorasick ac;
    for (int i = 0; i < n; ++i) {
        string s;
        cin >> s;
        ac.insert(s);
    }

    ac.build();

    string t;
    cin >> t;

    cout << ac.find_cnt(t) << "\n";

    return 0;
}

```

// test problem: <https://www.luogu.com.cn/problem/P3808>

6.2 exkmp.cpp

```

#include <cstdio>
#include <cstring>
#include <iostream>
#include <string>

using namespace std;
const int K = 100005;
int nt[K], extand[K];
char S[K], T[K];
void Getnext(char *T, int *next) {
    int len = strlen(T), a = 0;
    next[0] = len;
    while (a < len - 1 && T[a] == T[a + 1])
        a++;
    next[1] = a;
    a = 1;
    for (int k = 2; k < len; k++) {

```

```

    int p = a + next[a] - 1, L = next[k - a];
    if ((k - 1) + L >= p) {
        int j = (p - k + 1) > 0 ? (p - k + 1) : 0;
        while (k + j < len && T[k + j] == T[j])
            j++;
        next[k] = j;
        a = k;
    } else
        next[k] = L;
}
}

void GetExand(char *S, char *T, int *next) {
    Getnext(T, next);
    int slen = strlen(S), tlen = strlen(T), a = 0;
    int MinLen = slen < tlen ? slen : tlen;
    while (a < MinLen && S[a] == T[a])
        a++;
    extand[0] = a;
    a = 0;
    for (int k = 1; k < slen; k++) {
        int p = a + extand[a] - 1, L = next[k - a];
        if ((k - 1) + L >= p) {
            int j = (p - k + 1) > 0 ? (p - k + 1) : 0;
            while (k + j < slen && j < tlen && S[k + j] == T[j])
                j++;
            extand[k] = j;
            a = k;
        } else
            extand[k] = L;
    }
}

int main() {
    while (scanf("%s%s", S, T) == 2) {
        GetExand(S, T, nt);
        for (int i = 0; i < strlen(T); i++)
            printf("%d ", nt[i]);
        puts("");
        for (int i = 0; i < strlen(S); i++)
            printf("%d ", extand[i]);
        puts("");
    }
}

```

6.3 kmp.cpp

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

```
using namespace std;
using ll = long long;
```

```

vector<int> prefixFunction(string s) {
    int n = (int)s.size();
    vector<int> p(n);
    for (int i = 1; i < n; ++i) {
        int j = p[i - 1];
        while (j > 0 && s[i] != s[j]) j = p[j - 1];
        if (s[i] == s[j]) ++j;
        p[i] = j;
    }
    return p;
}

```

```

// KMP based on prefixFunction. return all match position in t
// also can create string st = s + '#' + t, and call prefixFunction(st),
// if p[i] == s.length() it's a successful match: s in t
vector<int> kmp(string s, string t) {
    vector<int> ans;

```

```

    int n = (int)s.size(), m = (int)t.size();
    if (n > m) return ans;
    auto p = prefixFunction(s);
    for (int i = 0, j = 0; i < m; ++i) {
        while (j > 0 && s[j] != t[i]) j = p[j - 1];
        if (s[j] == t[i] && ++j == n) ans.emplace_back(i - n + 1);
    }
    return ans;
}

```

```

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string t, s;
    cin >> t >> s;

    string st = s + '#' + t;
    auto ans = prefixFunction(st);
    for (int i = s.length() + 1; i < st.length(); ++i) {
        if (ans[i] == s.length()) {
            cout << i - 2 * s.length() + 1 << "\n";
        }
    }

    for (int i = 0; i < s.length(); ++i) {
        cout << ans[i] << " \n"[i == s.length() - 1];
    }

    return 0;
}

```

// test problem: <https://www.luogu.com.cn/problem/P3375>

6.4 manacher.cpp

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

```
using namespace std;
using ll = long long;
```

```

template <typename T>
vector<int> manacher(int n, const T &s) {
    if (n == 0) {
        return vector<int>();
    }
    vector<int> res(2 * n - 1, 0);
    int l = -1, r = -1;
    for (int z = 0; z < 2 * n - 1; z++) {
        int i = (z + 1) >> 1;
        int j = z >> 1;
        int p = (i >= r ? 0 : min(r - i, res[2 * (l + r) - z]));
        while (j + p + 1 < n && i - p - 1 >= 0) {
            if (!s[j + p + 1] == s[i - p - 1]) {
                break;
            }
            p++;
        }
        if (j + p > r) {
            l = i - p;
            r = j + p;
        }
        res[z] = p;
    }
    return res;
}
// res[2 * i] = odd radius in position i
// res[2 * i + 1] = even radius between positions i and i + 1

```

```

// s = "abaa" -> res = {0, 0, 1, 0, 0, 1, 0}
// s = "aaa" -> res = {0, 1, 1, 1, 0}
// in other words, for every z from 0 to 2 * n - 2:
// calculate i = (z + 1) >> 1 and j = z >> 1
// now there is a palindrome from i - res[z] to j + res[z]
// (watch out for i > j and res[z] = 0)
}
template <typename T>
vector<int> manacher(const T &s) {
    return manacher((int)s.size(), s);
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string s;
    cin >> s;
    int n = s.length();

    auto ans = manacher(s);

    int len = 0, id = -1;
    for (int z = 0; z < 2 * n - 1; ++z) {
        if (z % 2 == 0 && 1 + 2 * ans[z] > len) { // odd length of palindrome
            len = 1 + 2 * ans[z];
            id = z / 2 - ans[z];
        } else if (z % 2 == 1 && 2 * ans[z] > len) { // even length of palindrome
            len = 2 * ans[z];
            id = z / 2 - ans[z] + 1;
        }
    }

    cout << s.substr(id, len) << "\n";

    return 0;
}

```

6.5 后缀数组.cpp

```

#include <cstdio>
#include <iostream>
#include <cstdlib>
#include <cstring>
using namespace std;
typedef long long ll;
const int N = 2e5 + 10;
int n, mx, mn;
int a[N];
char s[N];
int SA[N], rnk[N], height[N], sum[N], tp[N];
//rnk[i]    μi,μ, SA[i]    II    iμμ, Height[i]    II    Iμ(i-1)I    μμLCP
//sum[i]    »,    iμ, tp[i]    rnkμ~(L    °~μ'),    SA

bool cmp(int *f, int x, int y, int w)
{
    return f[x] == f[y] && f[x + w] == f[y + w];
}

void get_SA(char *s, int n, int m)
{
    //    I    °~μ1μ
    for (int i = 0; i < m; i++)
        sum[i] = 0; // 0
    for (int i = 0; i < n; i++)
        sum[rnk[i] = s[i]]++; //    J    %ȳ,³
    for (int i = 1; i < m; i++)

```

```

        sum[i] += sum[i - 1]; //sum[i]    L iμ
    for (int i = n - 1; i >= 0; i--)
        SA[--sum[rnk[i]]] = i; //    ±0ȳ°,
    //SA[i]    I    iμμ±,SA[--sum[rnk[i]]] = i    II    I%~±iμ--sum[rnk[i]],
    for (int len = 1; len <= n; len <= 1)
    {
        int p = 0;
        //    %SA¹
        for (int i = n - len; i < n; i++)
            tp[p++] = i; //    °i,ū¹, I    %~μ¹ȳ°,
        for (int i = 0; i < n; i++)
        {
            if (SA[i] >= len)
                tp[p++] = SA[i] - len;
        }
        //tp[i]    °~μ¹iμ±
        //    L    %¹~μ¹, I    °π%1μ
        for (int i = 0; i < m; i++)
            sum[i] = 0;
        for (int i = 0; i < n; i++)
            sum[rnk[tp[i]]]++;
        for (int i = 1; i < m; i++)
            sum[i] += sum[i - 1];
        for (int i = n - 1; i >= 0; i--)
            SA[--sum[rnk[tp[i]]]] = tp[i];
        //    %SA°rnk%rnk
        swap(rnk, tp); //    %>>>°tprnk
        p = 1;
        rnk[SA[0]] = 0;
        for (int i = 1; i < n; i++)
        {
            rnk[SA[i]] = cmp(tp, SA[i - 1], SA[i], len) ? p - 1 : p++; //    °rnk[i]
            //rnk[i]    -i]
        }
        if (p >= n)
            break;
        m = p; //
    }

    // height
    int k = 0;
    n--;
    for (int i = 0; i <= n; i++)
        rnk[SA[i]] = i;
    for (int i = 0; i < n; i++)
    {
        if (k)
            k--;
        int j = SA[rnk[i] - 1];
        while (s[i + k] == s[j + k])
            k++;
        height[rnk[i]] = k;
    }

    void check()
    {
        //    getchar();//    °ūō
        scanf("%s", s);
        int n = strlen(s);
        get_SA(s, n + 1, 'z' + 1);
        ll res = 0;
        for (int i = 1; i <= n; ++i)
            res += n - SA[i] - height[i];
        printf("%lld\n", res);
    }

    //    °Ł~°μ-

```



```
int main()
{
    int t;
    scanf("%d", &t);
    while (t--)
        check();
}
```

7 dp

7.1 数位dp.cpp

```
#include <algorithm>
#include <cstring>
#include <iostream>
#include <map>
using namespace std;
typedef long long ll;
int a, b, num[20], dp[20][2];
int dfs(int len, bool if6, bool lim) {
    if (len == 0) return 1;
    if (!lim && dp[len][if6]) return dp[len][if6];
    int cnt = 0, maxx = (lim ? num[len] : 9);
    for (int i = 0; i <= maxx; i++) {
        if (i == 4 || (if6 && i == 2)) continue;
        cnt += dfs(len - 1, i == 6, lim && i == maxx);
    }
    return lim ? cnt : dp[len][if6] = cnt;
}

int solve(int x) {
    memset(num, 0, sizeof(num));
    int k = 0;
    while (x) {
        num[++k] = x % 10;
        x /= 10;
    }
    return dfs(k, false, true);
}

int main() {
    scanf("%d%d", &a, &b);
    printf("%d\n", solve(b) - solve(a - 1));
}
```

7.2 最长上升子序列.cpp

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

using namespace std;
using ll = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
    }

    //
    vector<int> dp(n, 1e9), pre(n);
    for (int i = 0; i < n; ++i) {
```

```
        *upper_bound(dp.begin(), dp.end(), a[i]) = a[i];
        pre[i] = lower_bound(dp.begin(), dp.end(), 1e9) - dp.begin();
    }

    int ans = *max_element(pre.begin(), pre.end());

    cout << ans << "\n";

    return 0;
}
```