Contents

1 Basic

1.1 Default Code [d41d8c]

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

using namespace std;
using ll = long long;

const int Mod = 1E9 + 7;
int add(int
      a, int b) { a += b; if (a >= Mod) a -= Mod; return a; }
int sub
     (int a, int b) { a -= b; if (a < 0) a += Mod; return a; }
int mul(int a, int b) { return 1LL * a * b % Mod; }
int power(int a, ll b) {
    int ans = 1;
    for (; b > 0; b >>= 1, a = mul(a, a))
        if (b & 1) ans = mul(ans, a);
    return ans;
}

void solve() {

}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    auto s = chrono::high_resolution_clock::now();
    int t = 1;
    cin >> t;
    while (t--) {
        solve();
    }
    auto e = chrono::high_resolution_clock::now();
    cerr << chrono::duration_cast
         <chrono::milliseconds>(e - s).count() << " ms\n";
    return 0;
}
```

1.2 Debug [d781c5]

```
CODE1="a"
CODE2="ac"
set -e
g++ $CODE1.cpp -o $CODE1
g++ $CODE2.cpp -o $CODE2
for ((i=0;;i++))
do
    echo "$i"
    g++ gen.cpp -o gen
    ./gen > input
    # python3 gen.py > input
    ./$CODE1 < input > $CODE1.out
    ./$CODE2 < input > $CODE2.out
    cmp $CODE1.out $CODE2.out || break
done
```

1.3 Compare Fuction [d41d8c]

```
// 1. sort, 二分搜刻在函式內 lambda 就好
// 2. priority queue 小到大是 >, set 是 <
// 3. set 不能 = , multiset 必須 =
// 4. 確保每個成員都要比到
// 5. pbds_multiset 不要用 lower_bound
// 6. 如果要用 find, 插入 inf 後使用 upper_bound
// 7. multiset 可以跟 set 一樣使用, 但請注意第 3、4 點
auto cmp = [](int i, int j) { return i > j; };
priority_queue<int, vector<int>, decltype(cmp)> pq(cmp);

vector<int> a {1, 2, 5, 4, 3}; // 小心不要改到 a
auto cmp = [&a](int i, int j) { return a[i] > a[j]; };
priority_queue<int, vector<int>, decltype(cmp)> pq(cmp);
```

1.4 Pbds [d41d8c]

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template<class T>
using pbds_set = tree<T, null_type,
    less<T>, rb_tree_tag, tree_order_statistics_node_update>;
template<class T>
using pbds_multiset = tree<T, null_type, less_equal
    <T>, rb_tree_tag, tree_order_statistics_node_update>;
```

1.5 Int128 [85923a]

```
using i128 = __int128_t; // 1.7E38
istream &operator>>(istream &is, i128 &a) {
    i128 sgn = 1; a = 0;
    string s; is >> s;
    for (auto c : s) {
        if (c == '-') {
            sgn = -1;
        } else {
            a = a * 10 + c - '0';
        }
    }
    a *= sgn;
    return is;
}
ostream &operator<<(ostream &os, i128 a) {
    string res;
    if (a < 0) os << '-', a = -a;
    while (a) {
        res.push_back(a % 10 + '0');
        a /= 10;
    }
    reverse(res.begin(), res.end());
    os << res;
    return os;
}
```

1.6 Rng [401544]

```
mt19937_64 rng
    (chrono::steady_clock::now().time_since_epoch().count());
ll x = rng();
shuffle(a.begin(), a.end(), rng);
```

2 Graph

2.1 Prim [cefbbf]

```
auto prim =
     [&](int n, vector<vector<pair<int, int>>> &adj) -> bool {
    int sz = 0; ll ans = 0;
    priority_queue<pair<int, int>,
         vector<pair<int, int>>, greater<pair<int, int>>> pq;
    pq.emplace(0, 0); // w, vertex
    vector<bool> vis(n);
    while (!pq.empty()) {
        auto [w, u] = pq.top(); pq.pop();
        if (vis[u]) continue;
        vis[u] = true;
        ans += w, sz++;
        for (auto [v, w] : g[u])
            if (!vis[v])
                pq.emplace(w, v);
    }
    if (sz == n) return true;
    return false;
};
```

2.2 Bellman-Ford [430de2]

```
// 用 Bellman Ford 找負環
void bellmanFord() {
    int n, m; cin >> n >> m;
    vector<array<int, 3>> e;
    for (int i = 0; i < m; i++) {
        int u, v, w; cin >> u >> v >> w;
        u--, v--; e.push_back({u, v, w});
    }
    vector<ll> dis(n, inf), par(n);
    int t = -1; dis[0] = 0;
    for (int i = 1; i <= n; i++) {
        for (auto [u, v, w] : e) {
            if (dis[v] > dis[u] + w) {
                dis[v] = dis[u] + w;
                par[v] = u;
                if (i == n) t = v;
            }
        }
    }
    if (t == -1) { cout << "NO\n"; return; }
    for (int i = 1; i < n; i++) t = par[t];
    vector<int> ans {t};
    int i = t;
    do {
        i = par[i];
        ans.push_back(i);
    } while (i != t);
    reverse(ans.begin(), ans.end());
    cout << "YES\n";
    for (auto x : ans) cout << x + 1 << " ";
}
```

2.3 Floyd-Warshall [db13dd]

```
const ll inf = 1E18;
void floydWarshall(int n, int m) {
    int n, m; cin >> n >> m;
    vector<vector<int>> dis(n, vector<int>(n, inf));
    for (int i = 0; i < m; i++) {
        int u, v, w; cin >> u >> v >> w;
        dis[u][v] = min(dis[u][v], w);
        dis[v][u] = min(dis[v][u], w);
    }
    for (int i = 0; i < n; i++) dis[i][i] = 0;
    for (int k = 0; k < n; k++)
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                dis[i][j
                    ] = min(dis[i][j], dis[i][k] + dis[k][j]);
}
```

```
const int N = 500; // Floyd 封包
void floyd(int n, vector<bitset<N>> &dp) {
    for (int k = 0; k < n; k++)
        for (int i = 0; i < n; i++)
            if (dp[i][k]) dp[i] |= dp[k];
}
```

2.4 Euler [4177dc]

```
// 1. 無向圖是歐拉圖：
// 非零度頂點是連通的
// 頂點的度數都是偶數

// 2. 無向圖是半歐拉圖(有路沒有環)：
// 非零度頂點是連通的
// 恰有 2 個奇度頂點

// 3. 有向圖是歐拉圖：
// 非零度頂點是強連通的
// 每個頂點的入度和出度相等

// 4. 有向圖是半歐拉圖(有路沒有環)：
// 非零度頂點是弱連通的
// 至多一個頂點的出度與入度之差為 1
// 至多一個頂點的入度與出度之差為 1
// 其他頂點的入度和出度相等
vector<int> ans;
auto dfs = [&](auto &&self, int u) -> void {
    while (g[u].size()) {
        int v = *g[u].begin();
        g[u].erase(v);
        self(self, v);
    }
    ans.push_back(u);
};
dfs(dfs, 0);
reverse(ans.begin(), ans.end());
```

2.5 DSU [6bd5f4]

```
struct DSU {
    int n;
    vector<int> f, siz;
    DSU(int n) : n(n), f(n), siz(n, 1) {
        iota(f.begin(), f.end(), 0);
    }
    int find(int x) {
        if (f[x] == x) return x;
        return f[x] = find(f[x]);
    }
    bool same(int x, int y) {
        return find(x) == find(y);
    }
    bool merge(int x, int y) {
        x = find(x); y = find(y);
        if (x == y) return false;
        if (siz[x] < siz[y]) swap(x, y);
        siz[x] += siz[y];
        f[y] = x;
        n--;
        return true;
    }
    int size(int x) {
        return siz[find(x)];
    }
};
struct DSU {
    int n;
    vector<int> f, siz, stk;
    DSU(int n) : n(n), f(n), siz(n, 1) {
        iota(f.begin(), f.end(), 0);
        stk.clear();
    }
    int find(int x) {
        return x == f[x] ? x : find(f[x]);
    }
    bool same(int x, int y) {
        return find(x) == find(y);
    }
    bool merge(int x, int y) {
        x = find(x); y = find(y);
        if (x == y) return false;
        if (siz[x] < siz[y]) swap(x, y);
        siz[x] += siz[y];
        f[y] = x;
        n--;
        stk.push_back(y);
        return true;
    }
    void undo(int x) {
        while (stk.size() > x) {
            int y = stk.back();
            stk.pop_back();
            n++;
            siz[f[y]] -= siz[y];
            f[y] = y;
        }
    }
    int size(int x) {
        return siz[find(x)];
    }
};
```

2.6 SCC [3ac1cb]

```
struct SCC {
    int n, cur, cnt;
    vector<vector<int>> adj;
    vector<int> stk, dfn, low, bel;
    SCC(int n) : n(n), cur
        (0), cnt(0), adj(n), dfn(n, -1), low(n), bel(n, -1) {}
    void addEdge(int u, int v) { adj[u].push_back(v); }
    void dfs(int x) {
        dfn[x] = low[x] = cur++;
        stk.push_back(x);
        for (auto y : adj[x]) {
            if (dfn[y] == -1) {
                dfs(y);
                low[x] = min(low[x], low[y]);
            } else if (bel[y] == -1) {
                low[x] = min(low[x], dfn[y]);
            }
        }
        if (dfn[x] == low[x]) {
            int y;
            do {
                y = stk.back();
                bel[y] = cnt;
                stk.pop_back();
            } while (y != x);
            cnt++;
        }
    }
    vector<int> work() {
        for (int i = 0; i < n; i++)
            if (dfn[i] == -1) dfs(i);
        return bel;
    }
    struct Graph {
        int n;
        vector<pair<int, int>> edges;
        vector<int> siz, cnte;
    };
    Graph compress() {
        Graph g;
        g.n = cnt;
        g.siz.resize(cnt);
        g.cnte.resize(cnt);
        for (int i = 0; i < n; i++) {
            g.siz[bel[i]]++;
            for (auto j : adj[i]) {
                if (bel[i] != bel[j]) {
                    g.edges.emplace_back(bel[i], bel[j]);
                } else {
                    g.cnte[bel[i]]++;
                }
            }
        }
        return g;
    }
};
```

2.7 VBCC [95997d]

```
struct VBCC {
    int n, cur, cnt;
    vector<vector<int>> adj, bcc;
    vector<int> stk, dfn, low;
    vector<bool> ap;
    VBCC(int n) : n(n), cur(0)
         , cnt(0), adj(n), bcc(n), ap(n), low(n), dfn(n, -1) {}
    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    void dfs(int x, int p) {
        dfn[x] = low[x] = cur++;
        stk.push_back(x);
        int ch = 0;
        for (auto y : adj[x]) {
            if (y == p) continue;
            if (dfn[y] == -1) {
                dfs(y, x), ch++;
                low[x] = min(low[x], low[y]);
                if (low[y] >= dfn[x]) {
                    int v;
                    do {
                        v = stk.back();
                        bcc[v].push_back(cnt);
                        stk.pop_back();
                    } while (v != y);
                    bcc[x].push_back(cnt);
                    cnt++;
                }
                if (low[y] >= dfn[x] && p != -1)
                    ap[x] = true;
            } else {
                low[x] = min(low[x], dfn[y]);
            }
        }
        if (p == -1 && ch > 1) ap[x] = true;
    }
    vector<bool> work() {
        for (int i = 0; i < n; i++)
            if (dfn[i] == -1) dfs(i, -1);
        return ap;
    }
    struct Graph {
        int n;
        vector<pair<int, int>> edges;
        vector<int> bel, siz, cnte;
    };
    Graph compress() {
        Graph g; // 壓完是一棵樹，但不一定每個 bel 都有節點
        g.bel.resize(n);
        g.siz.resize(n);
        g.cnte.resize(n);
        for (int u = 0; u < n; u++) {
            if (ap[u]) {
                g.bel[u] = cnt++;
                g.siz.emplace_back();
                g.cnte.emplace_back();
                for (auto v : bcc[u]) {
                    g.edges.emplace_back(g.bel[u], v);
                }
            } else if (bcc[u].size() == 1) {
                g.bel[u] = bcc[u][0];
            }
            g.siz[g.bel[u]]++;
        }
        g.n = cnt;
        for (int i = 0; i < n; i++)
            for (auto j : adj[i])
                if (g.bel[i] == g.bel[j] && i < j)
                    g.cnte[g.bel[i]]++;
        return g;
    }
};
```

2.8 EBCC [12a170]

```
struct EBCC { // CF/contest/1986/pF
    int n, cur, cnt;
    vector<vector<int>> adj;
    vector<int> stk, dfn, low, bel;
    vector<pair<int, int>> bridges; // 關鍵邊
    EBCC(int n) : n(n), cur
         (0), cnt(0), adj(n), low(n), dfn(n, -1), bel(n, -1) {}
    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    void dfs(int x, int p) {
        dfn[x] = low[x] = cur++;
        stk.push_back(x);
        for (auto y : adj[x]) {
            if (y == p) continue;
            if (dfn[y] == -1) {
                dfs(y, x);
                low[x] = min(low[x], low[y]);
                if (low[y] > dfn[x]) {
                    bridges.emplace_back(x, y);
                }
            } else if (bel[y] == -1) {
                low[x] = min(low[x], dfn[y]);
            }
        }
        if (dfn[x] == low[x]) {
            int y;
            do {
                y = stk.back();
                bel[y] = cnt;
                stk.pop_back();
            } while (y != x);
            cnt++;
        }
    }
    vector<int> work() { // not connected
        for (int i = 0; i < n; i++)
            if (dfn[i] == -1) dfs(i, -1);
        return bel;
    }
    struct Graph {
        int n;
        vector<pair<int, int>> edges;
        vector<int> siz, cnte;
    };
    Graph compress() {
        Graph g;
        g.n = cnt;
        g.siz.resize(cnt);
        g.cnte.resize(cnt);
        for (int i = 0; i < n; i++) {
            g.siz[bel[i]]++;
            for (auto j : adj[i]) {
                if (bel[i] < bel[j]) {
                    g.edges.emplace_back(bel[i], bel[j]);
                } else if (i < j) {
                    g.cnte[bel[i]]++;
                }
            }
        }
        return g;
    }
};
```

2.9 2-SAT [28688f]

```
struct TwoSat {
    int n; vector<vector<int>> e;
    vector<bool> ans;
    TwoSat(int n) : n(n), e(2 * n), ans(n) {}
    void addClause(int u, bool f, int v, bool g) {
        e[2 * u + !f].push_back(2 * v + g);
        e[2 * v + !g].push_back(2 * u + f);
    }
    void ifThen(int u, bool f, int v, bool g) {
        // 必取 A: not A -> A
        e[2 * u + !f].push_back(2 * v + g);
    }
    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 : e[u]) {
                if (dfn[v] == -1) {
                    tarjan(v);
                    low[u] = min(low[u], low[v]);
                } else if (id[v] == -1) { // in stk
                    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; }
};
```

2.10 Functional Graph [c314e3]

```
const int N = 2E5;
const int Lg = __lg(N); // __lg(max(n, qi)), [0, Lg]
int cht[N][Lg];
struct FuntionalGraph {
    int n, cnt;
    vector<int> g, bel, id, cycsz, in, top, hei;
    FuntionalGraph(const vector<int> &g) : n(g.size()), cnt(0)
         , g(g), bel(n, -1), id(n), in(n), top(n, -1), hei(n) {
        for (int i = 0; i < n; i++)
            cht[i][0] = g[i], in[g[i]]++;
        for (int i = 1; i <= Lg; i++)
            for (int u = 0; u < n; u++) {
                int nxt = cht[u][i - 1];
                cht[u][i] = cht[nxt][i - 1];
            }
        for (int i = 0; i < n; i++)
            if (in[i] == 0) label(i);
        for (int i = 0; i < n; i++)
            if (top[i] == -1) label(i);
    }
    void label(int u) {
        vector<int> p; int cur = u;
        while (top[cur] == -1) {
            top[cur] = u;
            p.push_back(cur);
            cur = g[cur];
        }
        auto s = find(p.begin(), p.end(), cur);
        vector<int> cyc(s, p.end());
        p.erase(s, p.end()); p.push_back(cur);
        for (int i = 0; i < (int)cyc.size(); i++)
            bel[cyc[i]] =
                 cnt, id[cyc[i]] = i, hei[cyc[i]] = cyc.size();
        if (!cyc.empty())
            ++cnt, cycsz.push_back(cyc.size());
        for (int i = p.size() - 1; i > 0; i--)
            id[p[i - 1]]
                 = id[p[i]] - 1, hei[p[i - 1]] = hei[p[i]] + 1;
    }
    int jump(int u, int k) {
        for (int b = 0; k > 0; b++) {
            if (k & 1) u = cht[u][b];
            k >>= 1;
        }
        return u;
    }
};
```

3 Data Structure

3.1 Segment Tree [d41d8c]

```
template<class Info, class Tag = bool()>
struct SegmentTree { // [l, r), uncomment /**/ to lazy
    int n;
    vector<Info> info;
    /*
    vector<Tag> tag;
    */
    template<class T>
    SegmentTree(const vector<T> &init) {
        n = init.size();
        info.assign(4 << __lg(n), Info());
        /*
        tag.assign(4 << __lg(n), Tag());
        */
        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] = info[2 * p] + info[2 * p + 1];
    }
    /*
    void apply(int p, int l, int r, const Tag &v) {
        info[p].apply(l, r, v);
        tag[p].apply(v);
    }
    void push(int p, int l, int r) {
        int m = (l + r) / 2;
        if (r - l >= 1) {
            apply(2 * p, l, m, tag[p]);
            apply(2 * p + 1, m, r, 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, l, r);
        */
        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 &i) {
        modify(1, 0, n, p, i);
    }
    Info query(int p, int l, int r, int ql, int qr) {
        if (qr <= l || ql >= r) return Info();
        if (ql <= l && r <= qr) return info[p];
        int m = (l + r) / 2;
        /*
        push(p, l, r);
        */
        return query(2 *
             p, l, m, ql, qr) + query(2 * p + 1, m, r, ql, qr);
    }
    Info query(int ql, int qr) {
        return query(1, 0, n, ql, qr);
    }
    /*
    void rangeApply
         (int p, int l, int r, int ql, int qr, const Tag &v) {
        if (qr <= l || ql >= r) return;
        if (ql <= l && r <= qr) {
            apply(p, l, r, v);
            return;
        }
        int m = (l + r) / 2;
        push(p, l, r);
        rangeApply(2 * p, l, m, ql, qr, v);
        rangeApply(2 * p + 1, m, r, ql, qr, v);
        pull(p);
    }
    void rangeApply(int l, int r, const Tag &v) {
        rangeApply(1, 0, n, l, r, v);
    }
    */
    template<class F>   // 尋找區間內，第一個符合條件的
    int findFirst
         (int p, int l, int r, int x, int y, F &&pred) {
        if (l >= y || r <= x) return -1;
        if (l >= x && r <= y && !pred(info[p])) return -1;
        if (r - l == 1) return l;
        int m = (l + r) / 2;
        /*
        push(p, l, r);
        */
        int res = findFirst(2 * p, l, m, x, y, pred);
        if (res == -1)
            res = findFirst(2 * p + 1, m, r, x, y, pred);
        return res;
    }
    template<class F>   // 若要找 last，先右子樹遞迴即可
    int findFirst(int l, int r, F &&pred) {
        return findFirst(1, 0, n, l, r, pred);
    }
};
// 有些 Tag 不用 push 例如 sweepLine
/*
struct Tag {
    int setVal = 0;
    int add = 0;
    void apply(const Tag &t) & {
        if (t.setVal) {
            setVal = t.setVal;
            add = t.add;
        } else {
            add += t.add;
        }
    }
};
*/
struct Info {
    ll sum = 0;
    /*
    void apply(int l, int r, const Tag &t) & {
        if (t.setVal) {
            sum = (r - l) * t.setVal;
        }
        sum += (r - l) * t.add;
    }
    */
    // 部分 assignment 使用
    // Info &operator=(const Info &rhs) & {
    //     return *this;
    // }
    Info &operator=(const ll &rhs) & {
        sum = rhs;
        return *this;
    }
};
Info operator+(const Info &a, const Info &b) {
    Info c;
    c.sum = a.sum + b.sum;
    return c;
}
```

3.2 Persistent Segment Tree [d41d8c]

```
template<class Info>
struct PST {
    struct Node {
        Info info = Info();
        int lc = 0, rc = 0;
    };
    int n;
    vector<Node> nd;
    vector<int> rt;
    template<class T>
    PST(const vector<T> &init) {
        n = init.size();
        nd.assign(1, Node());
        rt.clear();
        function<int(int, int)> build = [&](int l, int r) {
            int id = nd.size();
            nd.emplace_back();
            if (r - l == 1) {
                nd[id].info = init[l];
                return id;
            }
            int m = (l + r) >> 1;
            nd[id].lc = build(l, m);
            nd[id].rc = build(m, r);
            pull(nd[id]);
            return id;
        };
        rt.push_back(build(0, n));
    }
    void pull(Node &t) {
        t.info = nd[t.lc].info + nd[t.rc].info;
    }
    int copy(int t) { // copy 一個 node
        nd.push_back(nd[t]);
        return nd.size() - 1;
    }
    int generate() { // 創立新的 node
        nd.emplace_back();
        return nd.size() - 1;
    }
    int modify(int t, int l, int r, int x, const Info &v) {
        t = t ? copy(t) : generate();
        if (r - l == 1) {
            nd[t].info = v;
            return t;
        }
        int m = (l + r) / 2;
        if (x < m) {
            nd[t].lc = modify(nd[t].lc, l, m, x, v);
        } else {
            nd[t].rc = modify(nd[t].rc, m, r, x, v);
        }
        pull(nd[t]);
        return t;
    }
    void modify(int ver, int p, const Info &i) {
        if (int(rt.size()) <= ver) rt.resize(ver + 1);
        rt[ver] = modify(rt[ver], 0, n, p, i);
    }
    Info query(int t, int l, int r, int ql, int qr) {
        if (l >= qr || r <= ql) return Info();
        if (ql <= l && r <= qr) return nd[t].info;
        int m = (l + r) / 2;
        return query(nd[t].
             lc, l, m, ql, qr) + query(nd[t].rc, m, r, ql, qr);
    }
    Info query(int ver, int ql, int qr) {
        return query(rt[ver], 0, n, ql, qr);
    }
    void createVersion(int ori_ver) {
        rt.push_back(copy(rt[ori_ver]));
    }
    void reserve(int n, int q) {
        nd.reserve(n + q * (2 * __lg(n) + 1));
        rt.reserve(q + 1);
    }
    void resize(int n) { rt.resize(n); }
};
struct Info {
    ll sum = 0;
};
Info operator+(const Info &a, const Info &b) {
    return { a.sum + b.sum };
}
```

3.3 Static Kth-element [d41d8c]

```
template<class T>
struct StaticKth : PST<int> {
    int dct(T x) {
        return lower_bound(s.begin(), s.end(), x) - s.begin();
    }
    vector<T> v, s; // array, sorted
    map<T, int> cnt;
    StaticKth(const vector<T> &v_) {
        s = v = v_;
        sort(s.begin(), s.end());
        s.resize(unique(s.begin(), s.end()) - s.begin());
        init(s.size());
        for (int i = 0; i < v.size(); i++) {
            createVersion(i);
            int d = dct(v[i]);
            modify(i + 1, d, ++cnt[d]);
        }
    }
    int work(int a, int b, int l, int r, int k) {
        if (r - l == 1) return l;
        int x = nd[nd[b].lc].info - nd[nd[a].lc].info;
        int m = (l + r) / 2;
        if (x >= k) {
            return work(nd[a].lc, nd[b].lc, l, m, k);
        } else {
            return work(nd[a].rc, nd[b].rc, m, r, k - x);
        }
    }
    int work(int l, int r, int k) { // [l, r), k > 0
        return s[work(rt[l], rt[r], 0, n, k)];
    }
};
```

3.4 Dynamic Kth-element [d41d8c]

```
// Fenwick(rt-indexed) 包線段樹
template<class T>
struct DynamicKth : PST<int> {
    int dct(T x) {
        return lower_bound(s.begin(), s.end(), x) - s.begin();
    }
    vector<T> v, s; // array, sorted
    DynamicKth(const vector<T> &v_, const vector<T> &s_)
        : PST<int>(vector<int>(s_.size(), 0)) {
        assert(is_sorted(s_.begin(), s_.end()));
        v = v_, s = s_;
        rt.resize(v.size());
        for (int
              i = 0; i < v.size(); i++) add(i, dct(v[i]), 1);
    }
    int modify(int t, int l, int r, int x, int v) {
        t = t ? t : generate();
        if (r - l == 1) {
            nd[t].info += v;
            return t;
        }
        int m = (l + r) / 2;
        if (x < m) {
            nd[t].lc = modify(nd[t].lc, l, m, x, v);
        } else {
            nd[t].rc = modify(nd[t].rc, m, r, x, v);
        }
        pull(nd[t]);
        return t;
    }
    void add(int p, int x, int val) {
        for (int i = p + 1; i <= rt.size(); i += i & -i)
            rt[i - 1] = modify(rt[i - 1], 0, s.size(), x, val);
    }
    void modify(int p, int y) {
        add(p, dct(v[p]), -1);
        v[p] = y;
        add(p, dct(v[p]), 1);
    }
    int work(
         vector<int> &a, vector<int> &b, int l, int r, int k) {
        if (r - l == 1) return l;
        int m = (l + r) / 2;
        int res = 0;
        for (auto x : a) res -= nd[nd[x].lc].info;
        for (auto x : b) res += nd[nd[x].lc].info;
        if (res >= k) {
            for (auto &x : a) x = nd[x].lc;
            for (auto &x : b) x = nd[x].lc;
            return work(a, b, l, m, k);
        } else {
            for (auto &x : a) x = nd[x].rc;
            for (auto &x : b) x = nd[x].rc;
            return work(a, b, m, r, k - res);
        }
    }
    int work(int l, int r, int k) { // [l, r), k > 0
        vector<int> a, b;
        for (int i = l; i > 0; i -= i & -i)
            a.push_back(rt[i - 1]);
        for (int i = r; i > 0; i -= i & -i)
            b.push_back(rt[i - 1]);
        return s[work(a, b, 0, s.size(), k)];
    }
};
```

3.5 Fenwick [d41d8c]

```
template<class T>
struct Fenwick {
    int n; vector<T> a;
    Fenwick(int n) : n(n), a(n) {}
    void add(int x, const T &v) {
        for (int i = x + 1; i <= n; i += i & -i)
            a[i - 1] = a[i - 1] + v;
    }
    T sum(int x) {
        T ans{};
        for (int i = x; i > 0; i -= i & -i)
            ans = ans + a[i - 1];
        return ans;
    }
    T rangeSum(int l, int r) {
        return sum(r) - sum(l);
    }
    int select(const T &k, int start = 0) {
        // 找到最小的 x, 使得 sum(x + 1) - sum(start) > k
        int x = 0; T cur = -sum(start);
        for (int i = 1 << __lg(n); i; i /= 2) {
            if (x + i <= n && cur + a[x + i - 1] <= k) {
                x += i;
                cur = cur + a[x - 1];
            }
        }
        return x;
    }
};
template<class T>
struct TwoDFenwick {
    int nx, ny;  // row, col 個數
    vector<vector<T>> a;
    TwoDFenwick(int nx, int ny) : nx(nx), ny(ny) {
        a.assign(nx, vector<T>(ny, T{}));
    }
    void add(int x, int y, const T &v) {
        for (int i = x + 1; i <= nx; i += i & -i)
            for (int j = y + 1; j <= ny; j += j & -j)
                a[i - 1][j - 1] = a[i - 1][j - 1] + v;
    }
    T sum(int x, int y) {
        T ans{};
        for (int i = x; i > 0; i -= i & -i)
            for (int j = y; j > 0; j -= j & -j)
                ans = ans + a[i - 1][j - 1];
        return ans;
    }
    T rangeSum(int lx, int ly, int rx, int ry) {
        return sum(
             rx, ry) - sum(lx, ry) - sum(rx, ly) + sum(lx, ly);
    }
};
```

3.6 Range Fenwick [d41d8c]

```
template<class T>
struct RangeFenwick { // 全部以 0 based 使用
    int n;
    vector<T> d, di;
    RangeFenwick(int n) : n(n), d(n), di(n) {}
    void add(int x, const T &v) {
        T vi = v * (x + 1);
        for (int i = x + 1; i <= n; i += i & -i) {
            d[i - 1] = d[i - 1] + v;
            di[i - 1] = di[i - 1] + vi;
        }
    }
    void rangeAdd(int l, int r, const T &v) {
        add(l, v); add(r, -v);
    }
    T sum(int x) { // 左閉右開查詢
        T ans{};
        for (int i = x; i > 0; i -= i & -i) {
            ans = ans + T(x + 1) * d[i - 1];
            ans = ans - di[i - 1];
        }
        return ans;
    }
    T rangeSum(int l, int r) { // 左閉右開查詢
        return sum(r) - sum(l);
    }
    int select(const T &k, int start = 0) {
        // 找到最小的 x, 使得 sum(x + 1) - sum(start) > k
        int x = 0; T cur = -sum(start);
        for (int i = 1 << __lg(n); i; i /= 2) {
            if (x + i <= n) {
                T val = T(
                     x + i + 1) * d[x + i - 1] - di[x + i - 1];
                if (cur + val <= k) {
                    x += i;
                    cur = cur + val;
                }
            }
        }
        return x;
    }
};
template<class T>
struct RangeTwoDFenwick {  // 全部以 0 based 使用
    int nx, ny;  // row, col 個數
    vector<vector<T>> d, di, dj, dij;
    RangeTwoDFenwick(int x, int y) : nx(x), ny(y) {
        d.assign(nx, vector<T>(ny, T{}));
        di.assign(nx, vector<T>(ny, T{}));
        dj.assign(nx, vector<T>(ny, T{}));
        dij.assign(nx, vector<T>(ny, T{}));
    }
    void add(int x, int y, const T &v) {
        T vi = v * (x + 1);
        T vj = v * (y + 1);
        T vij = v * (x + 1) * (y + 1);
        for (int i = x + 1; i <= nx; i += i & -i) {
            for (int j = y + 1; j <= ny; j += j & -j) {
                d[i - 1][j - 1] = d[i - 1][j - 1] + v;
                di[i - 1][j - 1] = di[i - 1][j - 1] + vi;
                dj[i - 1][j - 1] = dj[i - 1][j - 1] + vj;
                dij[i - 1][j - 1] = dij[i - 1][j - 1] + vij;
            }
        }
    }
    void rangeAdd(int lx, int ly, int rx, int ry, const T &v) {
        add(rx, ry, v);
        add(lx, ry, -v);
        add(rx, ly, -v);
        add(lx, ly, v);
    }
    T sum(int x, int y) { // 左閉右開查詢
        T ans{};
        for (int i = x; i > 0; i -= i & -i) {
            for (int j = y; j > 0; j -= j & -j) {
                ans = ans
                      + T(x * y + x + y + 1) * d[i - 1][j - 1];
                ans = ans - T(y + 1) * di[i - 1][j - 1];
                ans = ans - T(x + 1) * dj[i - 1][j - 1];
                ans = ans + dij[i - 1][j - 1];
            }
        }
        return ans;
    }
    T rangeSum
         (int lx, int ly, int rx, int ry) { // 左閉右開查詢
        return sum(
             rx, ry) - sum(lx, ry) - sum(rx, ly) + sum(lx, ly);
    }
};
```

3.7 KDTree [d41d8c]

```
struct Info {
    static constexpr int DIM = 2;
    array<int, DIM> x, L, R;
    int v = 0, sum = 0;
    void pull(const Info &l, const Info &r) {
        sum = v + l.sum + r.sum;
    }
};
struct KDTree {
    static constexpr int DIM = Info::DIM;
    vector<Info> info;
    vector<int> rt, l, r, p;
    int n = 0, lg;
    KDTree(int n) : info(1), lg(__lg(n)), l(n + 1), r(n + 1) {
        rt.resize(lg + 1);
    }
    void pull(int p) {
        info[p].L = info[p].R = info[p].x;
        info[p].pull(info[l[p]], info[r[p]]);
        for (int ch : {l[p], r[p]}) {
            if (!ch) continue;
            for (int k = 0; k < DIM; k++) {
                info[p
                     ].L[k] = min(info[p].L[k], info[ch].L[k]);
                info[p
                     ].R[k] = max(info[p].R[k], info[ch].R[k]);
            }
        }
    }
    int rebuild(int l, int r, int dep = 0) {
        if (r == l) return 0;
        int m = (l + r) / 2;
        nth_element
             (p.begin() + l, p.begin() + m, p.begin() + r,
            [&](int x, int y)
                  { return info[x].x[dep] < info[y].x[dep]; });
        int x = p[m];
        this->l[x] = rebuild(l, m, (dep + 1) % DIM);
        this->r[x] = rebuild(m + 1, r, (dep + 1) % DIM);
        pull(x);
        return x;
    }
    void append(int &x) {
        if (!x) return;
        p.push_back(x);
        append(l[x]);
        append(r[x]);
        x = 0;
    }
    void addNode(const Info &i) {
        p.assign(1, info.size());
        info.push_back(i);
        for (int j = 0;; j++) {
            if (!rt[j]) {
                rt[j] = rebuild(0, p.size());
                break;
            } else {
                append(rt[j]);
            }
        }
    }
    Info query(int p,
          const array<int, DIM> &l, const array<int, DIM> &r) {
        if (!p) return Info();
        bool inside = true;
        for (int k = 0; k < DIM; k++) {
            inside &= (
                 l[k] <= info[p].L[k] && info[p].R[k] <= r[k]);
        }
        if (inside) return info[p];
        for (int k = 0; k < DIM; k++) {
            if (info[p].R[k] < l[k] || r[k] < info[p].L[k]) {
                return Info();
            }
        }
        Info ans;
        inside = true;
        for (int k = 0; k < DIM; k++) {
            inside &=
                  l[k] <= info[p].x[k] && info[p].x[k] <= r[k];
        }
        if (inside) ans = info[p];
        ans.pull(
             query(this->l[p], l, r), query(this->r[p], l, r));
        return ans;
    }
    Info query
        (const array<int, DIM> &l, const array<int, DIM> &r) {
        Info res;
        for (int i = 0; i <= lg; i++) {
            res.pull(res, query(rt[i], l, r));
        }
        return res;
    }
};
```

3.8 Treap [d41d8c]

```
template<class Info, class Tag = bool()>
struct Treap { // 0 -> initial root
    vector<Info> info;
    // vector<Tag> tag;
    vector<int> siz, par, rev, pri;
    vector<array<int, 2>> ch;
    Treap(int n) : info(n + 1), siz(n
          + 1), par(n + 1), rev(n + 1), pri(n + 1), ch(n + 1) {
        // tag.resize(n + 1);
        for (int i = 1; i <= n; i++)
            siz[i] = 1, pri[i] = rand();
    }
    // void apply(int t, const Tag &v) {
    //     info[t].apply(siz[t], v);
    //     tag[t].apply(v);
    // }
    void push(int t) {
        if (rev[t]) {
            swap(ch[t][0], ch[t][1]);
            if (ch[t][0]) rev[ch[t][0]] ^= 1;
            if (ch[t][1]) rev[ch[t][1]] ^= 1;
            rev[t] = 0;
        }
        // apply(ch[t][0], tag[t]);
        // apply(ch[t][1], tag[t]);
        // tag[t] = Tag();
    }
    void pull(int t) {
        siz[t] = 1 + siz[ch[t][0]] + siz[ch[t][1]];
        info[t].pull(info[ch[t][0]], info[ch[t][1]]);
    }
    int merge(int a, int b) {
        if (!a || !b) return a ? a : b;
        push(a), push(b);
        if (pri[a] > pri[b]) {
            ch[a][1] = merge(ch[a][1], b);
            pull(a); return a;
        } else {
            ch[b][0] = merge(a, ch[b][0]);
            pull(b); return b;
        }
    }
    pair<int, int> split(int t, int k) {
        if (!t) return {0, 0};
        push(t);
        if (siz[ch[t][0]] >= k) {
            auto [a, b] = split(ch[t][0], k);
            ch[t][0] = b, pull(t);
            return {a, t};
        } else {
            auto [a
                 , b] = split(ch[t][1], k - siz[ch[t][0]] - 1);
            ch[t][1] = a, pull(t);
            return {t, b};
        }
    }
    template<class F>   // 尋找區間內，第一個符合條件的
    int findFirst(int t, F &&pred) {
        if (!t) return 0;
        push(t);
        if (!pred(info[t])) return 0;
        int idx = findFirst(ch[t][0], pred);
        if (!idx) idx
             = 1 + siz[ch[t][0]] + findFirst(ch[t][1], pred);
        return idx;
    }
    int getPos(int rt, int t) { // get t's index in array
        int res = siz[t] + 1;
        while (t != rt) {
            int p = par[t];
            if (ch[p][1] == t) res += siz[ch[p][0]] + 1;
            t = p;
        }
        return res;
    }
    void getArray(int t, vector<Info> &a) {
        if (!t) return;
        push(t);
        getArray(ch[t][0], a);
        a.push_back(info[t]);
        getArray(ch[t][1], a);
    }
};
struct Tag {
    int setVal; ll add;
    void apply(const Tag &t) {
        if (t.setVal) {
            setVal = t.setVal;
            add = t.add;
        } else {
            add += t.add;
        }
    }
};
struct Info {
    ll val, sum;
    void apply(int siz, const Tag &t) {
        if (t.setVal) {
            val = t.setVal;
            sum = 1LL * siz * t.setVal;
        }
        val += t.add;
        sum += 1LL * siz * t.add;
    }
    void pull(const Info &l, const Info &r) {
        sum = val + l.sum + r.sum;
    }
};
```

3.9 RMQ [d41d8c]

```
template<class T, class F = less<T>>
struct RMQ { // [l, r)
    int n;
    F cmp = F();
    vector<vector<T>> g;
    RMQ() {}
    RMQ(const vector<T> &a, F cmp = F()) : cmp(cmp) {
        init(a);
    }
    void init(const vector<T> &a) {
        n = a.size();
        int lg = __lg(n);
        g.resize(lg + 1);
        g[0] = a;
        for (int j = 1; j <= lg; j++) {
            g[j].resize(n - (1 << j) + 1);
            for (int i = 0; i <= n - (1 << j); i++)
                g[j][i] = min(g[j -
                     1][i], g[j - 1][i + (1 << (j - 1))], cmp);
        }
    }
    T operator()(int l, int r) {
        assert(0 <= l && l < r && r <= n);
        int lg = __lg(r - l);
        return min(g[lg][l], g[lg][r - (1 << lg)], cmp);
    }
};
```

3.10 Mo [d41d8c]

```
struct Query { int id, l, r; };
void mo(vector<Query> &q) {
    int blk = sqrt(q.size());
    sort(q.begin
         (), q.end(), [&](const Query &a, const Query &b) {
        int x = a.l / blk, y = b.l / blk;
        return x == y ? a.r < b.r : x < y;
    });
}
int nl = 0, nr = -1;
for (auto [id, l, r] : qry) {
    while (nr < r) nr++, addR();
    while (l < nl) nl--, addL();
    while (r < nr) delR(), nr--;
    while (nl < l) delL(), nl++;
}
```

4 Flow Matching

4.1 Dinic [d41d8c]

```
template<class T>
struct Dinic {
    // argument time: O(VE), O(E) for unit capacity,
    // argument number: O(V), min(O(E^0.5), O(V^2/3)) for unit
          capacity, O(V^0.5) for deg_in(u) or deg_out(u) <= 1
    // so bipartite matching: O(EV^0.5)
    struct Edge {
        int to;
        T f, cap; // 流量跟容量
    };
    int n, m, s, t;
    const T INF_FlOW = numeric_limits<T>::max() / 2;
    vector<vector<int>> g;
    vector<Edge> e;
    vector<int> h, cur;
    Dinic(int n) : n(n), m(0), g(n), h(n), cur(n) {}
    void addEdge(int u, int v, T cap) {
        e.push_back({v, 0, cap});
        e.push_back({u, 0, 0});
        g[u].push_back(m++);
        g[v].push_back(m++);
    }
    bool bfs() {
        fill(h.begin(), h.end(), -1);
        h[s] = 0; queue<int> q;
        q.push(s);
        while (!q.empty()) {
            int u = q.front(); q.pop();
            for (int id : g[u]) {
                auto [v, f, cap] = e[id];
                if (f == cap) continue;
                if (h[v] == -1) {
                    h[v] = h[u] + 1;
                    if (v == t) return true;
                    q.push(v);
                }
            }
        }
        return false;
    }
    T dfs(int u, T flow) {
        if (flow == 0) return 0;
        if (u == t) return flow;
        for (int &i = cur[u]; i < g[u].size(); i++) {
            int j = g[u][i];
            auto [v, f, cap] = e[j];
            if (h[u] + 1 != h[v]) continue;
            if (f == cap) continue;
            T mn = dfs(v, min(flow, cap - f));
            if (mn > 0) {
                e[j].f += mn;
                e[j ^ 1].f -= mn;
                return mn;
            }
        }
        return 0;
    }
    T work(int s_, int t_) {
        s = s_; t = t_; T f = 0;
        while (bfs()) {
            fill(cur.begin(), cur.end(), 0);
            while (true) {
                T res = dfs(s, INF_FlOW);
                if (res == 0) break;
                f += res;
            }
        }
        return f;
    }
    void reuse(int n_) { // 走殘留網路，res += f
        while (n < n_) {
            g.emplace_back();
            h.emplace_back();
            cur.emplace_back();
            n += 1;
        }
    }
};
```

4.2 Min Cut [d41d8c]

```
void minCut(int n, int m, Dinic<int> d) {
    int ans = d.work(0, n - 1);
    vector<int> vis(n);
    auto dfs = [&](auto self, int u) -> void {
        if (vis[u]) continue;
        vis[u] = 1;
        for (int id : d.g[u]) {
            auto [to, f, cap] = d.e[id];
            if (cap - f > 0) self(self, to);
        }
    };
    dfs(dfs, 0);
    for (int i = 0; i < n; i++) {
        if (!vis[i]) continue;
        for (int id : d.g[i]) {
            if (id & 1) continue;
            auto e = d.e[id];
            if (!vis[e.to])
                cout << i + 1 << " " << e.to + 1 << "\n";
        }
    }
}
```

4.3 MCMF [d41d8c]

```
template<class Tf, class Tc>
struct MCMF {
    struct Edge {
        int to;
        Tf f, cap; // 流量跟容量
        Tc cost;
    };
    int n, m, s, t;
    const Tf INF_FLOW = numeric_limits<Tf>::max() / 2;
    const Tc INF_COST = numeric_limits<Tc>::max() / 2;
    vector<Edge> e;
    vector<vector<int>> g;
    vector<Tc> dis, pot;
    vector<int> rt, inq;
    MCMF(int n) : n(n), m(0), g(n) {}
    void addEdge(int u, int v, Tf cap, Tc cost) {
        e.push_back({v, 0, cap, cost});
        e.push_back({u, 0, 0, -cost});
        g[u].push_back(m++);
        g[v].push_back(m++);
    }
    bool spfa() {   // O(FVE)
        dis.assign(n, INF_COST);
        rt.assign(n, -1), inq.assign(n, 0);
        queue<int> q; q.push(s);
        dis[s] = 0, inq[s] = 1;
        while (!q.empty()) {
            int u = q.front(); q.pop();
            inq[u] = 0;
            for (int id : g[u]) {
                auto [v, f, cap, cost] = e[id];
                Tc ndis = dis[u] + cost + pot[u] - pot[v];
                if (f < cap && dis[v] > ndis) {
                    dis[v] = ndis, rt[v] = id;
                    if (!inq[v])
                        q.push(v), inq[v] = 1;
                }
            }
        }
        return dis[t] != INF_COST;
    }
    bool dijkstra() {   // O(FElogV)
        dis.assign(n, INF_COST), rt.assign(n, -1);
        priority_queue<pair<Tc, int>,
             vector<pair<Tc, int>>, greater<pair<Tc, int>>> pq;
        dis[s] = 0; pq.emplace(dis[s], s);
        while (!pq.empty()) {
            auto [d, u] = pq.top(); pq.pop();
            if (dis[u] < d) continue;
            for (int id : g[u]) {
                auto [v, f, cap, cost] = e[id];
                Tc ndis = dis[u] + cost + pot[u] - pot[v];
                if (f < cap && dis[v] > ndis) {
                    dis[v] = ndis, rt[v] = id;
                    pq.emplace(ndis, v);
                }
            }
        }
        return dis[t] != INF_COST;
    }
    pair<Tf, Tc> work(int s_, int t_, Tf need) {
        s = s_, t = t_; pot.assign(n, 0);
        Tf flow{}; Tc cost{}; int fr = 0;
        while (fr++ ? dijkstra() : spfa()) {
            for (int i = 0; i < n; i++)
                dis[i] += pot[i] - pot[s];
            Tf f = need;
            for (int i = t; i != s; i = e[rt[i] ^ 1].to)
                f = min(f, e[rt[i]].cap - e[rt[i]].f);
            for (int i = t; i != s; i = e[rt[i] ^ 1].to)
                e[rt[i]].f += f, e[rt[i] ^ 1].f -= f;
            flow += f, need -= f;
            cost += f * dis[t];
            swap(dis, pot);
            if (need == 0) break;
        }
        return {flow, cost};
    }
    void reset() {
        for (int i = 0; i < m; i++) e[i].f = 0;
    }
};
```

4.4 Hungarian [d41d8c]

```
struct Hungarian { // 0-based, O(VE)
    int n, m;
    vector<vector<int>> adj;
    vector<int> used, vis;
    vector<pair<int, int>> match;
    Hungarian(int n, int m) : n(n), m(m) {
        adj.assign(n + m, {});
        used.assign(n + m, -1);
        vis.assign(n + m, 0);
    }
    void addEdge(int u, int v) {
        adj[u].push_back(n + v);
        adj[n + v].push_back(u);
    }
    bool dfs(int u) {
        int sz = adj[u].size();
        for (int i = 0; i < sz; i++) {
            int v = adj[u][i];
            if (vis[v] == 0) {
                vis[v] = 1;
                if (used[v] == -1 || dfs(used[v])) {
                    used[v] = u;
                    return true;
                }
            }
        }
        return false;
    }
    vector<pair<int, int>> work() {
        match.clear();
        used.assign(n + m, -1);
        vis.assign(n + m, 0);
        for (int i = 0; i < n; i++) {
            fill(vis.begin(), vis.end(), 0);
            dfs(i);
        }
        for (int i = n; i < n + m; i++)
            if (used[i] != -1)
                match.emplace_back(used[i], i - n);
        return match;
    }
};
```

4.5 Theorem [d41d8c]

```
// 有向無環圖：
// 最小不相交路徑覆蓋：
// 最小路徑數 = 頂點數 - 最大匹配數
// 最小相交路徑覆蓋：
// 先用
     Floyd 求傳遞封包，有連邊就建邊，然後再套最小不相交路徑覆蓋
// 二分圖：
// 最小點
     覆蓋：選出一些點，讓所有邊至少有一個端點在點集中的最少數量
// 最小點覆蓋 = 最大匹配數
// 還原解，flow 的作法是從源點開始 dfs，只走 cap - flow > 0
// 的邊，最後挑選左邊還沒被跑過的點和右邊被跑過的點當作覆蓋的點
// 最少邊覆蓋：選出一些邊，讓所有點都覆蓋到的最少數量
// 最少邊覆蓋 = 點數 - 最大匹配數
// 最大獨立集：選出一些點，使這些點兩兩沒有邊連接的最大數量
// 最大獨立集 = 點數 - 最大匹配數
```

5 String

5.1 Hash [234076]

```
const int D = 59;
vector<int> rollingHash(string &s) {
    vector<int> a {0};
    for (auto c : s)
        a.push_back(mul(a.back(), D) + (c - 'A' + 1));
    return a;
}
int qryHash(vector<int> &h, int l, int r) { // [l, r)
    return sub(h[r], mul(h[l], power(D, r - l)));
}
```

5.2 KMP [e3717b]

```
struct KMP {
    string sub;
    vector<int> fail;
    // fail 存匹配失敗時，移去哪
    // 也就是 sub(0, i) 的最長共同前後綴長度
    // ex : a   b  c a b c
    //     -1 -1 -1 0 1 2
    KMP(const string &sub_) { build(sub_); }
    vector<int> build(const string &sub_) {
        sub = sub_, fail.resize(sub.size(), -1);
        for (int i = 1; i < sub.size(); i++) {
            int now = fail[i - 1];
            while (now != -1 && sub[now + 1] != sub[i])
                now = fail[now];
            if (sub[now + 1] == sub[i])
                fail[i] = now + 1;
        }
        return fail;
    }
    vector<int> match(const string &s) {
        vector<int> match;
        for (int i = 0, now = -1; i < s.size(); i++) {
            while (s[i] != sub[now + 1] && now != -1)
                now = fail[now];
            if (s[i] == sub[now + 1]) now++;
            if (now + 1 == sub.size()) {
                match.push_back(i - now);
                now = fail[now];
            }
        }
        return match;
    }
};
```

5.3 Z Function [5b63dc]

```
// z[i] 表示 s 和 s[i, n - 1] (以 s[i] 開頭的後綴)
// 的最長公共前綴 (LCP) 的長度
vector<int> Z(const string &s) {
    int n = s.size();
    vector<int> z(n);
    z[0] = n; // lcp(s, s), -1 or n
    for (int i = 1, j = 1; i < n; i++) {
        z[i] = max(0, min(j + z[j] - i, z[i - j]));
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) z[i]++;
        if (i + z[i] > j + z[j]) j = i;
    }
    return z;
}
```

5.4 Manacher [1eb30d]

```
// 找到對於每個位置的迴文半徑
vector<int> manacher(const string &s) {
    string t = "#";
    for (auto c : s) t = t + c + '#';
    int n = t.size();
    vector<int> r(n);
    for (int i = 0,
         j = 0; i < n; i++) { // i 是中心, j 是最長回文字串中心
        if (2 * j - i >= 0 && j + r[j] > i)
            r[i] = min(r[2 * j - i], j + r[j] - i);
        while (i - r[i] >=
              0 && i + r[i] < n && t[i - r[i]] == t[i + r[i]])
            r[i] += 1;
        if (i + r[i] > j + r[j]) j = i;
    }
    return r;
}
// # a # b # a #
// 1 2 1 4 1 2 1
// # a # b # b # a #
// 1 2 1 2 5 2 1 2 1
// 值 -1 代表原回文字串長度
// (id - val + 1) / 2 可得原字串回文開頭
```

5.5 Trie [6c7186]

```
const int N = 1E7; // 0 -> initial state
const int ALPHABET_SIZE = 26;
int tot = 0;
int trie[N][ALPHABET_SIZE], cnt[N];
void reset() {
    tot = 0, fill_n(trie[0], ALPHABET_SIZE, 0);
}
int newNode() {
    int x = ++tot;
    cnt[x] = 0, fill_n(trie[x], ALPHABET_SIZE, 0);
    return x;
}
void add(const string &s) {
    int p = 0;
    for (auto c : s) {
        int &q = trie[p][c - 'a'];
        if (!q) q = newNode();
        p = q;
    }
    cnt[p] += 1;
}
int find(const string &s) {
    int p = 0;
    for (auto c : s) {
        int q = trie[p][c - 'a'];
        if (!q) return 0;
        p = q;
    }
    return cnt[p];
}
```

5.6 SA [b04578]

```
struct SuffixArray {
    int n;
    vector<int> sa, rk, lc;
    // n: 字串長度
    // sa: 後綴數組，sa[i] 表示第 i 小的後綴的起始位置
    // rk: 排名數組，rk[i] 表示從位置 i 開始的後綴的排名
    // lc: LCP
         數組，lc[i] 表示 sa[i] 和 sa[i + 1] 的最長公共前綴長度
    SuffixArray(const string &s) {
        n = s.length();
        sa.resize(n);
        lc.resize(n - 1);
        rk.resize(n);
        iota(sa.begin(), sa.end(), 0);
        sort(sa.begin(), sa.
             end(), [&](int a, int b) { return s[a] < s[b]; });
        rk[sa[0]] = 0;
        for (int i = 1; i < n; i++)
            rk[sa[i]]
                  = rk[sa[i - 1]] + (s[sa[i]] != s[sa[i - 1]]);
        int k = 1;
        vector<int> tmp, cnt(n);
        tmp.reserve(n);
        while (rk[sa[n - 1]] < n - 1) {
            tmp.clear();
            for (int
                  i = 0; i < k; i++) tmp.push_back(n - k + i);
            for (auto i : sa) if (i >= k) tmp.push_back(i - k);
            fill(cnt.begin(), cnt.end(), 0);
            for (int i = 0; i < n; i++) cnt[rk[i]]++;
            for (int i = 1; i < n; i++) cnt[i] += cnt[i - 1];
            for (int i = n - 1;
                  i >= 0; i--) sa[--cnt[rk[tmp[i]]]] = tmp[i];
            swap(rk, tmp); rk[sa[0]] = 0;
            for (int
                 i = 1; i < n; i++) rk[sa[i]] = rk[sa[i - 1]] +
                  (tmp[sa[i - 1]] < tmp[sa[i]] || sa[i - 1] + k
                  == n || tmp[sa[i - 1] + k] < tmp[sa[i] + k]);
            k *= 2;
        }
        for (int i = 0, j = 0; i < n; i++) {
            if (rk[i] == 0) {
                j = 0;
            } else {
                for (j -=
                     j > 0; i + j < n && sa[rk[i] - 1] + j < n
                     && s[i + j] == s[sa[rk[i] - 1] + j]; j++);
                lc[rk[i] - 1] = j;
            }
        }
    }
};
RMQ<int> rmq(sa.lc);
auto lcp = [&](int i, int j) { // [i, j]
    i = sa.rk[i], j = sa.rk[j];
    if (i > j) swap(i, j);
    assert(i != j);
    return rmq(i, j);
};
```

5.7 SAM [50a2d0]

```
struct SAM {
    // 0 -> initial state
    static constexpr int ALPHABET_SIZE = 26;
    // node -> strings with the same endpos set
    // link -> longest suffix with different endpos set
    // len -> state's longest suffix
    // fpos -> first endpos
    // strlen range -> [len(link) + 1, len]
    struct Node {
        int len, link = -1, fpos;
        array<int, ALPHABET_SIZE> next;
    };
    vector<Node> t;
    SAM() : t(1) {}
    int newNode() {
        t.emplace_back();
        return t.size() - 1;
    }
    int extend(int p, int c) {
        int cur = newNode();
        t[cur].len = t[p].len + 1;
        t[cur].fpos = t[cur].len - 1;
        while (p != -1 && !t[p].next[c]) {
            t[p].next[c] = cur;
            p = t[p].link;
        }
        if (p == -1) {
            t[cur].link = 0;
        } else {
            int q = t[p].next[c];
            if (t[p].len + 1 == t[q].len) {
                t[cur].link = q;
            } else {
                int r = newNode();
                t[r] = t[q];
                t[r].len = t[p].len + 1;
                while (p != -1 && t[p].next[c] == q) {
                    t[p].next[c] = r;
                    p = t[p].link;
                }
                t[q].link = t[cur].link = r;
            }
        }
        return cur;
    }
};
void solve(int n, string s, ll k) { // Substring Order II
    vector<int> last(n + 1);
    SAM sam;
    for (int i = 0; i < n; i++)
        last[i + 1] = sam.extend(last[i], s[i] - 'a');
    int sz = sam.t.size();
    vector<int> cnt(sz); // endpos size
    for (int i = 1; i <= n; i++) cnt[last[i]]++;
    vector<vector<int>> g(sz);
    for (int i = 1; i < sz; i++)
        g[sam.t[i].link].push_back(i);
    auto dfs = [&](auto self, int u) -> void {
        for (auto v : g[u])
            self(self, v), cnt[u] += cnt[v];
    }; dfs(dfs, 0);

    vector<ll> dp(sz, -1);
    // for any path from root
         , how many substring's prefix is the the path string
    auto rec = [&](auto self, int u) -> ll {
        if (dp[u] != -1) return dp[u];
        dp[u] = cnt[u]; // distinct: = 1
        for (int c = 0; c < SAM::ALPHABET_SIZE; c++) {
            int v = sam.t[u].next[c];
            if (v) dp[u] += self(self, v);
        }
        return dp[u];
    };
    rec(rec, 0);

    int p = 0; string ans;
    while (k > 0) { // 1-based
        for (int c = 0; c < SAM::ALPHABET_SIZE; c++) {
            int v = sam.t[p].next[c];
            if (v) {
                if (k > dp[v]) {
                    k -= dp[v];
                } else {
                    ans.push_back('a' + c);
                    k -= cnt[v]; // distinct: --
                    p = v; break;
                }
            }
        }
    } cout << ans << "\n";
}
```

5.8 Palindrome Tree [e5a1ed]

```
struct PAM {
    // 0 -> even root, 1 -> odd root
    static constexpr int ALPHABET_SIZE = 26;
    // fail -> longest prefix(suffix) palindrome
    // number end at i = end at link[last[i]] + 1
    struct Node {
        int len, fail, cnt;
        array<int, ALPHABET_SIZE> next;
        Node() : len{}, fail{}, next{} {}
    };
    vector<int> s;
    vector<Node> t;
    PAM() {
        t.assign(2, Node());
        t[0].len = 0, t[0].fail = 1;
        t[1].len = -1;
    }
    int newNode() {
        t.emplace_back();
        return t.size() - 1;
    }
    int getFail(int p, int i) {
        while (i - t[p].len < 1 || s[i - t[p].len - 1] != s[i])
            p = t[p].fail;
        return p;
    }
    int extend(int p, int c) {
        int i = s.size();
        s.push_back(c);
        p = getFail(p, i);
        if (!t[p].next[c]) {
            int r = newNode();
            int v = getFail(t[p].fail, i);
            t[r].len = t[p].len + 2;
            t[r].fail = t[v].next[c];
            t[p].next[c] = r;
        }
        return p = t[p].next[c];
    }
};
void solve() {
    string s; cin >> s;
    int n = s.length();
    vector<int> last(n + 1);
    last[0] = 1;
    PAM pam;
    for (int i = 0; i < n; i++)
        last[i + 1] = pam.extend(last[i], s[i] - 'a');
    int sz = pam.t.size();
    vector<int> cnt(sz);
    for (int i = 1; i <= n; i++)
        cnt[last[i]]++; // 去重 = 1
    for (int i = sz - 1; i > 1; i--)
        cnt[pam.t[i].fail] += cnt[i];
}
```

5.9 Duval [aed467]

```
// duval_algorithm
// 將字串分解成若干個非嚴格遞減的非嚴格遞增字串
vector<string> duval(string s) {
    int i = 0, n = s.size();
    vector<string> res;
    while (i < n) {
        int k = i, j = i + 1;
        while (s[k] <= s[j] && j < n) {
            if (s[k] < s[j]) k = i;
            else k++;
            j++;
        }
        while (i <= k) {
            res.push_back(s.substr(i, j - k));
            i += j - k;
        }
    }
    return res;
}
// 最小旋轉字串
string minRound(string s) {
    s += s;
    int i = 0, n = s.size(), start = i;
    while (i < n / 2) {
        start = i;
        int k = i, j = i + 1;
        while (s[k] <= s[j] && j < n) {
            if (s[k] < s[j]) k = i;
            else k++;
            j++;
        }
        while (i <= k) i += j - k;
    }
    return s.substr(start, n / 2);
}
```

6 Math

6.1 Mint [49cc47]

```
ll mul(ll a, ll b, ll p) {
    ll res = a * b - ll(1.L * a * b / p) * p;
    res %= p;
    if (res < 0) res += p;
    return res;
}
// 改 MLong: getMod() < (1ULL << 31)，會爆用 mul
template<class T>
constexpr T power(T a, ll b) {
    T res {1};
    for (; b > 0; b >>= 1, a = a * a)
        if (b & 1) res = res * a;
    return res;
}
template<int P>
struct Mint {
    static int Mod;
    static int getMod()
    { return P > 0 ? P : Mod; }
    static void setMod(int Mod_)
    { Mod = Mod_; }
    ll x;
    Mint(ll x = 0) : x {norm(x % getMod())} {}
    ll norm(ll x) const {
        if (x < 0) x += getMod();
        if (x >= getMod()) x -= getMod();
        return x;
    }
    explicit operator int() const { return x; }
    Mint operator-() const
    { return Mint(norm(getMod() - x)); }
    Mint inv() const
    { return power(*this, getMod() - 2); }
    Mint operator+(Mint rhs) const
    { return Mint(norm(x + rhs.x)); }
    Mint operator-(Mint rhs) const
    { return Mint(norm(x - rhs.x)); }
    Mint operator*(Mint rhs) const
    { return Mint(mul(x, rhs.x, getMod())); }
    Mint operator/(Mint rhs) const
    { return *this * rhs.inv(); }


    Mint &operator+=(Mint rhs) { return *this = *this + rhs; }
    Mint &operator-=(Mint rhs) { return *this = *this - rhs; }
    Mint &operator*=(Mint rhs) { return *this = *this * rhs; }
    Mint &operator/=(Mint rhs) { return *this = *this / rhs; }

    friend istream &operator>>(istream &is, Mint &a)
    { ll v; is >> v; a = Mint(v); return is; }
    friend ostream &operator<<(ostream &os, Mint a)
    { return os << a.x; }
    bool operator==(Mint y) const { return x == y.x; }
    bool operator!=(Mint y) const { return x != y.x; }
};
template<>
int Mint<0>::Mod = 998244353;
constexpr int P = 1E9 + 7;
using Z = Mint<P>;
```

6.2 Combination [f12983]

```
// C(m, n) = C(m, n - 1) * (m - n + 1) / n
struct Comb {
    int n;
    vector<Z> _fac, _invfac, _inv;
    Comb() : n{0}, _fac{1}, _invfac{1}, _inv{0} {}
    Comb(int n) : Comb() { init(n); }
    void init(int m) {
        m = min(m, Z::getMod() - 1);
        if (m <= n) return;
        _fac.resize(m + 1);
        _invfac.resize(m + 1);
        _inv.resize(m + 1);
        for (int i = n + 1; i <= m; i++) {
            _fac[i] = _fac[i - 1] * i;
        }
        _invfac[m] = _fac[m].inv();
        for (int i = m; i > n; i--) {
            _invfac[i - 1] = _invfac[i] * i;
            _inv[i] = _invfac[i] * _fac[i - 1];
        }
        n = m;
    }
    Z fac(int m) {
        if (m > n) init(2 * m);
        return _fac[m];
    }
    Z invfac(int m) {
        if (m > n) init(2 * m);
        return _invfac[m];
    }
    Z inv(int m) {
        if (m > n) init(2 * m);
        return _inv[m];
    }
    Z binom(int n, int m) {
        if (n < m || m < 0) return 0;
        return fac(n) * invfac(m) * invfac(n - m);
    }
    Z lucas(int n, int m) { // Mod 要在 1E5 左右
        if (m == 0) return 1;
        return binom(n % Z::getMod(), m % Z::getMod()) *
            lucas(n / Z::getMod(), m / Z::getMod());
    }
} comb; // 若要換模數需重新宣告
```

6.3 Sieve [37ae54]

```
vector<int> primes, minp;
void sieve(int n) {
    minp.assign(n + 1, 0);
    primes.clear();
    // minp[i] == i, 質數
    for (int i = 2; i <= n; i++) {
        if (minp[i] == 0) {
            minp[i] = i;
            primes.push_back(i);
        }
        for (auto p : primes) {
            if (i * p > n) break;
            minp[i * p] = p;
            if (p == minp[i]) break;
        }
    }
}
// a ^ (m-1) = 1 (Mod m)
// a ^ (m-2) = 1/a (Mod m)
// Exp2: cout << power(x, power(y, p, Mod - 1), Mod)
// Num = (x+1) * (y+1) * (z+1)...
// Sum = (a^0 + a^1+...+ a^x) * (b^0 +...+ b^y)
// Mul = N * (x+1) * (y+1) * (z+1) / 2
```

6.4 Miller Rabin Pollard Rho [394cfb]

```
ll mul(ll a, ll b, ll p) {
    ll res = a * b - ll(1.L * a * b / p) * p;
    res %= p;
    if (res < 0) res += p;
    return res;
}
ll power(ll a, ll b, ll p) {
    ll res {1};
    for (; b; b /= 2, a = mul(a, a, p))
        if (b & 1) res = mul(res, a, p);
    return res;
}
vector<ll
    > chk {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
bool check(ll a, ll d, int s, ll n) {
    a = power(a, d, n);
    if (a <= 1) return 1;
    for (int i = 0; i < s; i++, a = mul(a, a, n)) {
        if (a == 1) return 0;
        if (a == n - 1) return 1;
    }
    return 0;
}
bool isPrime(ll n) {
    if (n < 2) return 0;
    if (n % 2 == 0) return n == 2;
    ll d = n - 1, s = 0;
    while (d % 2 == 0) d /= 2, s++;
    for (ll i : chk)
        if (!check(i, d, s, n)) return 0;
    return 1;
}
const vector<ll> small = {2, 3, 5, 7, 11, 13, 17, 19};
ll findFactor(ll n) {
    if (isPrime(n)) return 1;
    for (ll p : small)
        if (n % p == 0) return p;
    ll x, y = 2, d, t = 1;
    auto f = [&](ll a) {
        return (mul(a, a, n) + t) % n;
    };
    for (int l = 2; ; l *= 2) {
        x = y;
        int m = min(l, 32);
        for (int i = 0; i < l; i += m) {
            d = 1;
            for (int j = 0; j < m; j++)
                y = f(y), d = mul(d, abs(x - y), n);
            ll g = __gcd(d, n);
            if (g == n) {
                l = 1, y = 2, ++t;
                break;
            }
            if (g != 1) return g;
        }
    }
}
map<ll, int> res;
void pollardRho(ll n) {
    if (n == 1) return;
    if (isPrime(n)) {
        res[n]++;
        return;
    }
    ll d = findFactor(n);
    pollardRho(n / d), pollardRho(d);
}
```

6.5 CRT [1a7c6e]

```
// a * p.first + b * p.second = gcd(a, b)
pair<ll, ll> exgcd(ll a, ll b) {
    if (b == 0) return {1, 0};
    auto [y, x] = exgcd(b, a % b);
    return {x, y - (a / b) * x};
}
// smallest non-negative solution
using i128 = __int128_t;
pair<ll, ll> CRT(ll r1, ll m1, ll r2, ll m2) {
    ll g = __gcd(m1, m2);
    if ((r2 - r1) % g) return {-1, g};
    m1 /= g, m2 /= g;
    auto [p1, p2] = exgcd(m1, m2);
    i128 lcm = i128(m1) * m2 * g;
    i128 res = i128(p1) * (r2 - r1) * m1 + r1;
    return {(res % lcm + lcm) % lcm, lcm};
}
ll EXCRT(vector<pair<ll, ll>> a) {
    ll R = 0, M = 1;
    for (auto [r, m] : a) {
        auto [res, lcm] = CRT(R, M, r, m);
        if (res == -1) return -1;
        R = res, M = lcm;
    }
    return R;
}
```

6.6 Matrix [2856cb]

```
template<class T>
vector<vector<T>> operator*(
     const vector<vector<T>> &a, const vector<vector<T>> &b) {
    int n = a.size(), k = a[0].size(), m = b[0].size();
    assert(k == b.size());
    vector<vector<T>> res(n, vector<T>(m));
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            for (int l = 0; l < k; l++)
                res[i][j] += a[i][l] * b[l][j];
    return res;
}
template<class T>
vector<vector<T>> unit(int n) {
    vector<vector<T>> res(n, vector<T>(n));
    for (int i = 0; i < n; i++) res[i][i] = 1;
    return res;
}
template<class T>
vector<vector<T>> power(vector<vector<T>> a, ll b) {
    int n = a.size();
    assert(n == a[0].size());
    auto res = unit<T>(n);
    for (; b; b /= 2, a = a * a)
        if (b % 2) res = res * a;
    return res;
}
using Matrix = vector<vector<Z>>;
```

6.7 Mex [00904e]

```
template<class T>
T mex(vector<T> &v) {
    unordered_set<T> s;
    for (auto e : v) s.insert(e);
    for (T i = 0; ; i++)
        if (s.find(i) == s.end()) return i;
}
```

6.8 Game Theorem

- sg 值為 0 代表先手必敗
- 當前 sg 值 $=$ 可能的後繼狀態的 mex (例如拿一個或拿兩個, 就等於兩者的 sg 值 mex), 若有互相依賴就兩個後繼狀態 xor 當作一組 sg 值 (例如切開成兩半, 只算一次)
- 單組基礎 nim 的 sg 值為本身的原因: $f(0) = 0, f(1) = mex(f(0)) = 1, f(2) = mex(f(0), f(1)) = 2...$, 都是自己
- 多組賽局可以把 sg 值 xor 起來, 當成最後的 sg 值, nim 也是一樣, 且由於 xor 性質, 如果可以快速知道 $sg(1)g(2)...g(n)$, 就可以用 xor 性質處理不連續組合

6.9 Fraction [62f33d]

```
template<class T>
struct Fraction {
    T n, d;
    void reduce() {
        T g = gcd(abs(n), abs(d));
        n /= g, d /= g;
        if (d < 0) n = -n, d = -d;
    }
    Fraction(T n = 0, T d = 1) : n(n), d(d)
    {
        assert(d != 0);
        reduce();
    }
    Fraction(const string &str) {
        char slash;
        if (str.find('/') != -1) {
            string x = str.substr(0, str.find('/'));
            string y = str.substr(str.find('/') + 1);
            n = stoBigint(x), d = stoBigint(y);
        } else {
            n = stoBigint(str), d = 1;
        }
        Fraction(n, d);
    }
    Fraction operator+(Fraction rhs) const
    { return Fraction(n * rhs.d + rhs.n * d, d * rhs.d); }
    Fraction operator-(Fraction rhs) const
    { return Fraction(n * rhs.d - rhs.n * d, d * rhs.d); }
    Fraction operator*(Fraction rhs) const
    { return Fraction(n * rhs.n, d * rhs.d); }
    Fraction operator/(Fraction rhs) const {
        assert(rhs.n != 0);
        return Fraction(n * rhs.d, d * rhs.n);
    }
    friend istream &operator>>(istream &is, Fraction &f) {
        string s; is >> s;
        f = Fraction(s);
        return is;
    }
    friend
          ostream &operator<<(ostream &os, const Fraction &f) {
        if (f.d == 1) os << f.n;
        else os << f.n << "/" << f.d;
        return os;
    }
    bool operator==(Fraction b) const
    { return n * b.d == b.n * d; }
    bool operator!=(Fraction b) const
    { return n * b.d != b.n * d; }
    bool operator<(Fraction b) const
    { return n * b.d < b.n * d; }
};
```

6.10 Gaussian Elimination [5d1aa7]

```
// 找反矩陣
     就開 2n，右邊放單位矩陣, 做完檢查左半是不是單位, 回傳右半
// 0  : no solution
// -1 : infinity solution
// 1  : one solution
template<class T>
tuple<T,
     int, vector<T>> gaussianElimination(vector<vector<T>> a) {
    T det = 1;
    bool zeroDet = false;
    int n = a.size(), m = a[0].size(), rk = 0, sgn = 1;
    for (int c = 0; c < n; c++) {
        int p = -1;
        for (int r = rk; r < n; r++) {
            if (a[r][c] != 0) {
                p = r;
                break;
            }
        }
        if (p == -1) {
            zeroDet = true;
            continue;
        }
        if (p != rk) swap(a[rk], a[p]), sgn *= -1;
        det *= a[rk][c];
        T inv = 1 / a[rk][c];
        for (int j = c; j < m; j++) a[rk][j] *= inv;
        for (int r = 0; r < n; r++) {
            if (r == rk || a[r][c] == 0) continue;
            T fac = a[r][c];
            for (int j = c; j < m; j++)
                a[r][j] -= fac * a[rk][j];
        }
        rk++;
    }
    det = (zeroDet ? 0 : det * sgn);
    for (int r = rk; r < n; r++)
        if (a[r][m - 1] != 0) return {det, 0, {}};
    if (rk < n) return {det, -1, {}};
    vector<T> ans(n);
    for (int i = 0; i < n; i++) ans[i] = a[i][m - 1];
    return {det, 1, ans};
}
template<class T>
tuple<int, vector
     <T>, vector<vector<T>>> findBasis(vector<vector<T>> a) {
    int n = a.size(), m = a[0].size(), rk = 0;
    vector<int> pos(m - 1, -1);
    for (int c = 0; c < m - 1; c++) {
        int p = -1;
        for (int r = rk; r < n; r++) {
            if (a[r][c] != 0) {
                p = r;
                break;
            }
        }
        if (p == -1) continue;
        if (p != rk) swap(a[rk], a[p]);
        pos[c] = rk;
        T inv = 1 / a[rk][c];
        for (int j = c; j < m; j++) a[rk][j] *= inv;
        for (int r = 0; r < n; r++) {
            if (r == rk || a[r][c] == 0) continue;
            T fac = a[r][c];
            for (int j = c; j < m; j++)
                a[r][j] -= fac * a[rk][j];
        }
        rk++;
    }
    vector<T> sol(m - 1);
    vector<vector<T>> basis;
    for (int r = rk; r < n; r++)
        if (a[r][m - 1] != 0)
            return {-1, sol, basis};
    for (int c = 0; c < m - 1; c++)
        if (pos[c] != -1)
            sol[c] = a[pos[c]][m - 1];
    for (int c = 0; c < m - 1; c++)
        if (pos[c] == -1) {
            vector<T> v(m - 1);
            v[c] = 1;
            for (int j = 0; j < m - 1; j++)
                if (pos[j] != -1)
                    v[j] = -a[pos[j]][c];
            basis.push_back(v);
        }
    return {rk, sol, basis};
}
template<class T>
using Matrix = vector<vector<T>>;
```

6.11 Integer Partition [83bc9d]

```
// CSES_Sum_of_Divisors
const int Mod = 1E9 + 7;
const int inv_2 = 500000004;
// n / 1 * 1 + n / 2 * 2 + n / 3 * 3 + ... + n / n * n
void integerPartition() {
    ll ans = 0, n; cin >> n;
    for (ll l = 1, r; l <= n; l = r + 1) {
        r = n / (n / l);
        ll val = n / l; // n / l 到 n / r 一樣的值
        ll sum = (((l + r) % Mod)
            * ((r - l + 1) % Mod)) % Mod * inv_2; // l 加到 r
        val %= Mod; sum %= Mod;
        ans += val * sum;
        ans %= Mod;
    }
    cout << ans << "\n";
}
```

6.12 Mobius Theorem

- 數論分塊可以快速計算一些含有除法向下取整的和式，就是像 $\sum_{i=1}^{n} f(i)g(\left\lfloor \frac{n}{i} \right\rfloor)$ 的和式。當可以在 $O(1)$ 內計算 $f(r)-f(l)$ 或已經預處理出 f 的前綴和時，數論分塊就可以在 $O(\sqrt{n})$ 的時間內計算上述和式的值。
- 迪利克雷捲積 $h(x)=\sum_{d|x} f(d)g(\frac{x}{d})$
- 積性函數
  - 莫比烏斯函數
    1. 定義

$$\sum_{d|n}\mu(d)=\begin{cases}1 & \text{for } n=1\\ 0 & \text{for } n\neq 0\end{cases}$$

    2. μ 是常數函數 1 的反元素
       $\Rightarrow \mu*1=\epsilon$，$\epsilon(n)$只在$n=1$時為 1，其餘情況皆為 0。
  - ϕ 歐拉函數: x 以下與 x 互質的數量

$$\begin{aligned}\phi*1 &= \sum_{d|n}\phi(\frac{n}{d}) \text{ 質因數分解}\\ &= \sum_{i=0}^{c}\phi(p^i)\\ &= 1+p^0(p-1)+p^1(p-1)+...+p^{c-1}(p-1)\\ &= p^c\\ &= id\end{aligned}$$

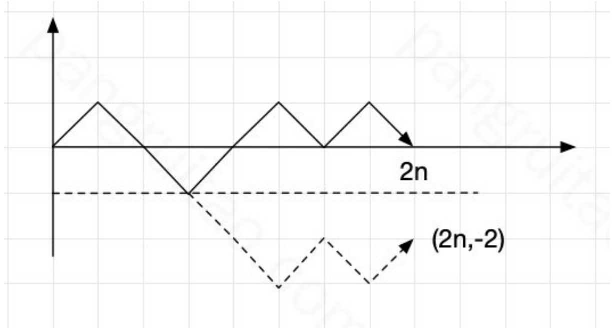
- 莫比烏斯反演公式
  - $f(n)=\sum_{d|n} g(d) \Leftrightarrow g(n)=\sum_{d|n}\mu(d)f(\frac{n}{d})$
  - $f(n)=\sum_{n|d} g(d) \Leftrightarrow g(n)=\sum_{n|d}\mu(\frac{d}{n})f(d)$
- 例子

$$\begin{aligned}&\sum_{i=a}^{b}\sum_{j=c}^{d}[gcd(i,j)=k]\\ &\Rightarrow \sum_{i=1}^{x}\sum_{j=1}^{y}[gcd(i,j)=k]\\ &= \sum_{i=1}^{\left\lfloor \frac{x}{k} \right\rfloor}\sum_{j=1}^{\left\lfloor \frac{y}{k} \right\rfloor}\epsilon(gcd(i,j))\\ &= \sum_{i=1}^{\left\lfloor \frac{x}{k} \right\rfloor}\sum_{j=1}^{\left\lfloor \frac{y}{k} \right\rfloor}\sum_{d|gcd(i,j)}\mu(d)\\ &= \sum_{d=1}^{\infty}\mu(d)\sum_{i=1}^{\left\lfloor \frac{x}{k} \right\rfloor}[d|i]\sum_{j=1}^{\left\lfloor \frac{y}{k} \right\rfloor}[d|j] \text{ d 可整除 i 時為 1}\\ &= \sum_{d=1}^{min(\left\lfloor \frac{x}{k} \right\rfloor,\left\lfloor \frac{y}{k} \right\rfloor)}\mu(d)\left\lfloor \frac{x}{kd} \right\rfloor\left\lfloor \frac{y}{kd} \right\rfloor\end{aligned}$$

6.13 Mobius Inverse [d41d8c]

```
const int N = 2E5;
ll pref[N];
void init() {
    pref[1] = 1;
    vector<ll>
          wei(N);  // wei = 0 代表是質數，-1 代表可被平方數整除
    for (ll i = 2; i < N; i++) {
        if (wei[i] == -1) {
            pref[i] = pref[i - 1];
            continue; // 包含平方
        }
        if (wei[i] == 0) {
            wei[i] = 1;
            for (ll j = 2; i * j < N; j++) {
                if (j % i == 0) wei[i * j] = -1;
                else if (wei[i * j] != -1) wei[i * j]++;
            }
        }
        pref[i] = pref[i - 1] + (wei[i] % 2 == 0 ? 1 : -1);
    }
}
void solve() {
    ll a, b, c, d, k; cin >> a >> b >> c >> d >> k;
    auto cal = [&](ll x, ll y) -> int {
        int res = 0;
        for (int l = 1, r; l <= min(x, y); l = r + 1) {
            r = min(x / (x / l), y / (y / l));
            res += (pref[r] - pref[l
                - 1]) * (x / l) * (y / l);  // 代推出來的式子
        }
        return res;
    };
    cout << cal
        (b / k, d / k) - cal((a - 1) / k, d / k) - cal(b / k,
         (c - 1) / k) + cal((a - 1) / k, (c - 1) / k) << "\n";
}
```

6.14 Catalan Theorem



1. n 個往上 n 個往下，先枚舉所有情況 $\frac{(2n)!}{n!n!} = C^{2n}_n$
2. 扣掉非法的，有多少種可能讓最後的點落在 $(2n,-2)$

假設往上有 x 個，往下有 y 個，會有:

$$\begin{cases} x+y=2n \\ y-x=2 \end{cases} \Rightarrow \begin{cases} x=n-1 \\ y=n+1 \end{cases}$$

所以只要扣掉 C^{2n}_{n-1} 即可

6.15 Burnside's Lemma

$|X/G| = \frac{1}{|G|}\sum_{g\in G}|X^g|$

- G：各種翻轉操作所構成的置換群
- X/G：本質不同的方案的集合
- X^g：對於某一種操作 g，所有方案中，經過 g 這種翻轉後保持不變的方案的集合
- 集合取絕對值代表集合數

7 Search and Gready

7.1 Binary Search [d41d8c]

```
void binarySearch() {
    // 二分找上界
    // 如果無解會 = 原 lo, lo 要先 - 1
    while (lo < hi) {
        int x = (lo + hi + 1) / 2;
        if (check(x)) lo = x;
        else hi = x - 1;
    }
    cout << lo;
    // 二分找下界
    // 如果無解會 = 原 hi, hi 要先 + 1
    while (lo < hi) {
        int x = (lo + hi) / 2;
        if (check(m)) hi = x;
        else lo = x + 1;
    }
    cout << lo;
}
```

7.2 Ternary Search [d41d8c]

```
void ternarySearch() {
    int lo = 0, hi = 10;
    while (lo < hi) {
        int xl = lo + (hi - lo) / 3;
        int xr = hi - (hi - lo) / 3;
        int resl = calc(xl), resr = calc(xr);
        if (resl < resr) {
            lo = xl + 1;
        } else {
            hi = xr - 1;
        }
    }
}
```

8 Tree

8.1 Binary Lifting LCA [fdf743]

```
const int N = 2E5;
const int Lg = __lg(N); // __lg(max(n, qi)), [0, Lg]
int up[N][Lg + 1];
vector<int> dep, dfn;
void build(int n, vector<vector<int>> &g, int rt = 0) {
    dep.assign(n, 0); dfn.assign(n, 0);
    int cur = 0;
    auto dfs = [&](auto self, int x, int p) -> void {
        dfn[x] = cur++;
        up[x][0] = p;
        for (int i = 1; i <= Lg; i++) {
            int nxt = up[x][i - 1];
            up[x][i] = up[nxt][i - 1];
        }
        for (auto y : g[x]) {
            if (y == p) continue;
            up[y][0] = x;
            dep[y] = dep[x] + 1;
            self(self, y, x);
        }
    };
    dfs(dfs, rt, rt);
}
int lca(int a, int b) {
    if (dep[a] < dep[b]) swap(a, b);
    int pull = dep[a] - dep[b];
    for (int i = 0; i <= Lg; i++)
        if (pull & (1 << i)) a = up[a][i];
    if (a == b) return a;
    for (int i = Lg; i >= 0; i--)
        if (up[a][i] != up[b][i])
            a = up[a][i], b = up[b][i];
    return up[a][0];
}
int jump(int x, int k) {
    for (int i = Lg; i >= 0; i--)
        if (k >> i & 1) x = up[x][i];
    return x;
}
int dist(int a, int b) {
    return dep[a] + dep[b] - 2 * dep[lca(a, b)];
}
```

8.2 Centroid Decomposition [2ecec4]

```
vector<bool> vis(n);
vector<int> siz(n), par(n, -1);
auto findSize = [&](auto self, int u, int p) -> int {
    siz[u] = 1;
    for (int v : g[u]) {
        if (v == p || vis[v]) continue;
        siz[u] += self(self, v, u);
    }
    return siz[u];
};
auto findCen = [&](auto self, int u, int p, int sz) -> int {
    for (int v : g[u]) {
        if (v == p || vis[v]) continue;
        if (siz[v] * 2 > sz) return self(self, v, u, sz);
    }
    return u;
};
auto buildCen = [&](auto self, int u, int p) -> void {
    findSize(findSize, u, p);
    int c = findCen(findCen, u, -1, siz[u]);
    vis[c] = true, par[c] = p;
    for (int v : g[c]) if (!vis[v]) self(self, v, c);
};
buildCen(buildCen, 0, -1);
```

8.3 Heavy Light Decomposition [9facc3]

```
struct HLD {
    int n, cur;
    vector<int> siz, top, dep, parent, in, out, seq;
    vector<vector<int>> adj;
    HLD(int n) : n(n), cur(0) {
        siz.resize(n); top.resize(n); dep.resize(n);
        parent.resize(n); in.resize(n); out.resize(n);
        seq.resize(n); adj.assign(n, {});
    }
    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    void work(int rt = 0) {
        top[rt] = rt;
        dep[rt] = 0;
        parent[rt] = -1;
        dfs1(rt); dfs2(rt);
    }
    void dfs1(int u) {
        if (parent[u] != -1)
            adj[u].erase(find
                (adj[u].begin(), adj[u].end(), parent[u]));
        siz[u] = 1;
        for (auto &v : adj[u]) {
            parent[v] = u, dep[v] = dep[u] + 1;
            dfs1(v);
            siz[u] += siz[v];
            if (siz[v] > siz[adj[u][0]]) {
                swap(v, adj[u][0]);
            } // 讓 adj[u][0] 是重子節點
        }
    }
    void dfs2(int u) {
        in[u] = cur++;
        seq[in[u]] = u; // dfn 對應的編號
        for (auto v : adj[u]) {
            top[v] = v == adj[u][0] ? top[u] : v;
            dfs2(v);
        }
        out[u] = cur;
    }
    int lca(int u, int v) {
        while (top[u] != top[v]) {
            if (dep[top[u]] > dep[top[v]]) {
                u = parent[top[u]];
            } else {
                v = parent[top[v]];
            }
        }
        return dep[u] < dep[v] ? u : v;
    }
    int dist(int u, int v) {
        return dep[u] + dep[v] - 2 * dep[lca(u, v)];
    }
    int jump(int u, int k) {
        if (dep[u] < k) return -1;
        int d = dep[u] - k;
        while (dep[top[u]] > d) u = parent[top[u]];
        return seq[in[u] - dep[u] + d];
    }
    bool isAncester(int u, int v) {
        return in[u] <= in[v] && in[v] < out[u];
    }
    int rootedParent(int rt, int v) {
        if (rt == v) return rt;
        if (!isAncester(v, rt)) return parent[v];
        auto it = upper_bound(adj[v].begin(), adj[v].end(), rt,
            [&](int x, int y) {
                return in[x] < in[y];
            }) - 1;
        return *it;
    }
    int rootedSize(int rt, int v) {
        if (rt == v) return n;
        if (!isAncester(v, rt)) return siz[v];
        return n - siz[rootedParent(rt, v)];
    }
    int rootedLca(int rt, int a, int b) {
        return lca(rt, a) ^ lca(a, b) ^ lca(b, rt);
    }
};
```

8.4 Link Cut Tree [544e55]

```
// 有用到 pathApply 才需要 apply 有關的
// 需要 pathQuery 才需要 pathInfo 有關的
// 需要 subtreeQuery 才需要 info, subtreeInfo
const int Mod = 51061;
struct Tag {
    ll add = 0, mul = 1;
    void apply(const Tag &v) {
        mul = mul * v.mul % Mod;
        add = (add * v.mul % Mod + v.add) % Mod;
    }
};
struct Info {
    int siz = 0;
    ll val = 0, sum = 0;
    void apply(const Tag &v) {
        val = (val * v.mul % Mod + v.add) % Mod;
        sum = (sum * v.mul % Mod + v.add * siz % Mod) % Mod;
    }
    void pull(const Info &l, const Info &r) {
        siz = 1 + l.siz + r.siz;
        sum = (l.sum + r.sum + val) % Mod;
    }
    Info &operator+=(const Info &i) {
        siz += i.siz;
        sum = (sum + i.sum) % Mod;
        return *this;
    }
    Info &operator-=(const Info &i) {
        siz -= i.siz;
        sum = (sum - (i.sum % Mod) + Mod) % Mod;
        return *this;
    }
};
struct LinkCutTree { // 1-based
    vector<Info> info, pathInfo, subtreeInfo;
    vector<Tag> tag;
    vector<array<int, 2>> ch;
    vector<int> p, rev;
    LinkCutTree
         (int n) : info(n + 1), pathInfo(n + 1), subtreeInfo(
         n + 1), tag(n + 1), ch(n + 1), p(n + 1), rev(n + 1) {}
    bool isrt(int x) {
        return ch[p[x]][0] != x && ch[p[x]][1] != x;
    }
    int pos(int x) { // x 是其 par 的左/右
        return ch[p[x]][1] == x;
    }
    void applyRev(int x) {
        swap(ch[x][0], ch[x][1]);
        rev[x] ^= 1;
    }
    void apply(int x, const Tag &v) {
        info[x].apply(v);
        pathInfo[x].apply(v);
        tag[x].apply(v);
    }
    void push(int x) {
        if (rev[x]) {
            if (ch[x][0]) applyRev(ch[x][0]);
            if (ch[x][1]) applyRev(ch[x][1]);
            rev[x] = 0;
        }
        if (ch[x][0]) apply(ch[x][0], tag[x]);
        if (ch[x][1]) apply(ch[x][1], tag[x]);
        tag[x] = Tag();
    }
    void pull(int x) {
        if (!x) return;
        pathInfo
             [x].pull(pathInfo[ch[x][0]], pathInfo[ch[x][1]]);
        info[x].pull(info[ch[x][0]], info[ch[x][1]]);
        info[x] += subtreeInfo[x];
    }
    void pushAll(int x) {
        if (!isrt(x)) pushAll(p[x]);
        push(x);
    }
    void rotate(int x) { // x 與其 par 交換位置
        int f = p[x], r = pos(x);
        ch[f][r] = ch[x][!r];
        if (ch[x][!r]) p[ch[x][!r]] = f;
        p[x] = p[f];
        if (!isrt(f)) ch[p[f]][pos(f)] = x;
        ch[x][!r] = f, p[f] = x;
        pull(f), pull(x);
    }
    void splay(int x) { // x 旋轉到當前的根
        pushAll(x);
        for (int f = p[x]; f = p[x], !isrt(x); rotate(x))
        if (!isrt(f)) rotate(pos(x) == pos(f) ? f : x);
    }
    // 第二次 access 可以回傳 LCA
    int access(int x) { // 根到 x 換成實鏈
        int c;
        for (c = 0; x; c = x, x = p[x]) {
            splay(x);
            subtreeInfo[x] += info[ch[x][1]];
            subtreeInfo[x] -= info[c];
            ch[x][1] = c;
            pull(x);
        }
        return c;
    }
    void makeRoot(int x) { // x 變成所在樹的根
        access(x), splay(x), applyRev(x);
    }
    int findRoot(int x) {
        access(x), splay(x);
        while (ch[x][0]) x = ch[x][0];
        splay(x); return x;
    }
    void split(int rt, int x) {
        makeRoot(x), access(rt), splay(rt);
    }
    void link(int rt, int x) {
        makeRoot(rt);
        access(x), splay(x);
        p[rt] = x;
        subtreeInfo[x] += info[rt];
        pull(x);
    }
    void cut(int rt, int x) {
        split(rt, x);
        ch[rt][0] = p[x] = 0;
        pull(rt);
    }
    bool connected(int x, int y) {
        return findRoot(x) == findRoot(y);
    }
    bool neighbor(int x, int y) {
        if (!connected(x, y)) return false;
        split(x, y);
        return pathInfo[x].siz == 2;
    }
    void modify(int x, const Info &v) {
        splay(x);
        info[x] = pathInfo[x] = v, pull(x);
    }
    void pathApply(int x, int y, const Tag &v) {
        assert(connected(x, y));
        split(x, y), apply(x, v);
    }
    Info pathQuery(int x, int y) {
        assert(connected(x, y));
        split(x, y); return pathInfo[x];
    }
    Info subtreeQuery(int rt, int x) {
        assert(connected(rt, x));
        split(rt, x);
        auto res = subtreeInfo[x];
        return res += pathQuery(x, x);
    }
};
```

8.5 Virtual Tree [c3a0b3]

```
// 多次詢問給某些關鍵點, 虛樹可達成快速樹 DP (前處理每個點)
// 例如這題是有權樹，給一些關鍵點, 求跟 vertex 1 隔開的最小成本
// 前處理 root 到所有點的最小邊權
vector<int> stk;
void insert(int key, vector<vector<int>> &vt) {
    if (stk.empty()) {
        stk.push_back(key);
        return;
    }
    int l = lca(stk.back(), key);
    if (l == stk.back()) {
        stk.push_back(key);
        return;
    }
    while (
         stk.size() > 1 && dfn[stk[stk.size() - 2]] > dfn[l]) {
        vt[stk[stk.size() - 2]].push_back(stk.back());
        stk.pop_back();
    }
    if (stk.size() < 2 || stk[stk.size() - 2] != l) {
        vt[l].push_back(stk.back());
        stk.back() = l;
    } else {
        vt[l].push_back(stk.back());
        stk.pop_back();
    }
    stk.push_back(key);
}
int work(vector<vector<int>> &vt) {
    while (stk.size() > 1) {
        vt[stk[stk.size() - 2]].push_back(stk.back());
        stk.pop_back();
    }
    int rt = stk[0];
    stk.clear();
    return rt;
}
void solve() {
    int n; cin >> n;
    vector<vector<int>> g(n);
    vector<vector<pair<int, int>>> wg(n);
    vector<vector<int>> vt(n);
    for (int i = 1; i < n; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        u--, v--;
        g[u].push_back(v), g[v].push_back(u);
        wg[u].emplace_back(v, w), wg[v].emplace_back(u, w);
    }
    build(n, g); // build LCA
    vector<int> dis(n, 1E9); // root 到各點的最小邊權
    auto dfs_dis = [&](auto &&self, int x, int p) -> void {
        for (auto [y, w] : wg[x]) {
            if (y == p) continue;
            dis[y] = min(w, dis[x]);
            self(self, y, x);
        }
    };
    dfs_dis(dfs_dis, 0, -1);

    vector<bool> isKey(n);
    vector<ll> dp(n);
    int q; cin >> q;
    while (q--) {
        int m; cin >> m;
        vector<int> key(m);
        for (int i = 0; i < m; i++) {
            cin >> key[i];
            key[i] -= 1;
            isKey[key[i]] = true;
        }
        key.push_back(0); // 固定 0 為 root, 看題目需求
        sort(key.begin(), key.end(), [&](int a, int b) {
            return dfn[a] < dfn[b];
        }); // 要 sort 再 insert
        for (auto x : key) insert(x, vt);
        work(vt);
        auto dfs = [&](auto &&self, int x) -> void {
            for (auto y : vt[x]) {
                self(self, y);
                if (isKey[y]) { // 直接砍了
                    dp[x] += dis[y];
                } else { // 不砍 or 砍
                    dp[x] += min<ll>(dp[y], dis[y]);
                } // 記得 reset
                isKey[y] = dp[y] = 0;
            }
            vt[x].clear(); // 記得 reset
        };
        dfs(dfs, 0);
        cout << dp[0] << "\n";
        dp[0] = 0; // 最後 reset root
    }
}
```

8.6 Dominator Tree [0cbb87]

```
// dom
     存起點到達此點的必經的上個節點(起點 = 自己), 無法到達 = -1
struct DominatorTree {
    int n, id;
    vector<vector<int>> adj, radj, bucket;
    vector<int> sdom, dom, vis, rev, pa, rt, mn, res;
    DominatorTree(int n) : n(n), id(0) {
        sdom.resize(n), rev.resize(n);
        pa.resize(n), rt.resize(n);
        mn.resize(n), res.resize(n);
        bucket.assign(n, {});
        adj.assign(n, {}), radj.assign(n, {});
        dom.assign(n, -1), vis.assign(n, -1);
    }
    void add_edge(int u, int v) { adj[u].push_back(v); }
    int query(int v, int x) {
        if (rt[v] == v) return x ? -1 : v;
        int p = query(rt[v], 1);
        if (p == -1) return x ? rt[v] : mn[v];
        if (sdom[mn[v]] > sdom[mn[rt[v]]])
            mn[v] = mn[rt[v]];
        rt[v] = p;
        return x ? p : mn[v];
    }
    void dfs(int v) {
        vis[v] = id, rev[id] = v;
        rt[id] = mn[id] = sdom[id] = id, id++;
        for (int u : adj[v]) {
            if (vis[u] == -1)
                dfs(u), pa[vis[u]] = vis[v];
            radj[vis[u]].push_back(vis[v]);
        }
    }
    vector<int> build(int s) {
        dfs(s);
        for (int i = id - 1; i >= 0; i--) {
            for (int u : radj[i])
                sdom[i] = min(sdom[i], sdom[query(u, 0)]);
            if (i) bucket[sdom[i]].push_back(i);
            for (int u : bucket[i]) {
                int p = query(u, 0);
                dom[u] = sdom[p] == i ? i : p;
            }
            if (i) rt[i] = pa[i];
        }
        res.assign(n, -1);
        for (int i = 1; i < id; i++)
            if (dom[i] != sdom[i])
                dom[i] = dom[dom[i]];
        for (int i = 1; i < id; i++)
            res[rev[i]] = rev[dom[i]];
        res[s] = s;
        for (int i = 0; i < n; i++)
            dom[i] = res[i];
        return dom;
    }
};
```

9 DP

9.1 LCS [9c3c7b]

```
string LCS(const string &a, const string &b) {
    int n = a.length(), m = b.length();
    vector<vector<int>> dp(n + 1, vector<int>(m + 1));
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            if (a[i - 1] == b[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
            }
        }
    }
    int l = dp[n][m];
    string ans(l, 0);
    while (n >= 1 && m >= 1) {
        if (a[n - 1] == b[m - 1]) {
            ans[l - 1] = a[n - 1];
            n--, m--, l--;
        } else {
            if (dp[n - 1][m] > dp[n][m - 1]) n--;
            else m--;
        }
    }
    return ans;
}
```

9.2 LIS [3018f4]

```
vector<int> LIS(const vector<int> &v) { // strictly
    int n = v.size(), L = 1;
    vector<int> dp(n); dp[0] = 1;
    vector<int> stk {v[0]};
    for (int i = 1; i < n; i++) {
        if (v[i] > stk.back()) { // >=
            stk.push_back(v[i]);
            dp[i] = ++L;
        } else { // upper
            auto it
                  = lower_bound(stk.begin(), stk.end(), v[i]);
            *it = v[i];
            dp[i] = it - stk.begin() + 1;
        }
    }
    vector<int> ans;
    for (int i = n - 1; i >= 0; i--)
        if (dp[i] == L) ans.push_back(v[i]), L--;
    reverse(ans.begin(), ans.end());
    return dp;
}
```

9.3 Edit Distance [b13609]

```
void editDistance() {
    string s1, s2; cin >> s1 >> s2;
    int n1 = s1.size(), n2 = s2.size();
    vector<int> dp(n2 + 1);
    iota(dp.begin(), dp.end(), 0);
    for (int i = 1; i <= n1; i++) {
        vector<int> cur(n2 + 1); cur[0] = i;
        for (int j = 1; j <= n2; j++) {
            if (s1[i - 1] == s2[j - 1]) {
                cur[j] = dp[j - 1];
            } else {
                // s1 新增等價於 s2 砍掉
                // dp[i][j] = min(s2 新增, 修改, s1 新增);
                cur[j]
                     = min({cur[j - 1], dp[j - 1], dp[j]}) + 1;
            }
        }
        swap(dp, cur);
    }
    cout << dp[n2] << "\n";
}
```

9.4 Bitmask [60bdb9]

```
void hamiltonianPath() {
    int n, m; cin >> n >> m;
    vector<vector<int>> adj(n);
    for (int i = 0; i < m; i++) {
        int u, v; cin >> u >> v;
        adj[--v].push_back(--u);
    }
    // 以...為終點，走過...
    vector dp(n, vector<int>(1 << n));
    dp[0][1] = 1;
    for (int mask = 1; mask < 1 << n; mask++) {
        if ((mask & 1) == 0) continue;
        for (int i = 0; i < n; i++) {
            if ((mask >> i & 1) == 0) continue;
            if (i == n - 1 && mask != (1 << n) - 1) continue;
            int pre = mask ^ (1 << i);
            for (int j : adj[i]) {
                if ((pre >> j & 1) == 0) continue;
                dp[i][mask] = (dp[i][mask] + dp[j][pre]) % Mod;
            }
        }
    }
    cout << dp[n - 1][(1 << n) - 1] << "\n";
}
void elevatorRides() {
    int n, x; cin >> n >> x;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];
    vector<int> dp(1 << n), f(1 << n);
    dp[0] = 1; // 次數、已使用人數
    for (int mask = 1; mask < 1 << n; mask++) {
        dp[mask] = 2E9;
        for (int i = 0; i < n; i++) {
            if ((mask >> i & 1) == 0) continue;
            int pre = mask ^ (1 << i);
            if (f[pre] + a[i] <= x) {
                if (dp[pre] < dp[mask] || dp[pre]
                      == dp[mask] && f[pre] + a[i] < f[mask]) {
                    dp[mask] = dp[pre];
                    f[mask] = f[pre] + a[i];
                }
            } else if (dp[pre] + 1 < dp[mask] ||
                  dp[pre] + 1 == dp[mask] && a[i] < f[mask]) {
                dp[mask] = dp[pre] + 1;
                f[mask] = a[i];
            }
        }
    }
    cout << dp[(1 << n) - 1] << "\n";
}
void minClique() { // 移掉一些邊，讓整張圖由最少團組成
    int n, m;
    cin >> n >> m;
    vector<bitset<N>> g(n);
    for (int i = 0; i < m; i++) {
        int u, v; cin >> u >> v;
        u--; v--; g[u][v] = g[v][u] = 1;
    }
    vector<int> dp(1 << n, inf);
    dp[0] = 1;
    for (int mask = 0; mask < 1 << n; mask++) { // 先正常 dp
        for (int i = 0; i < n; i++) {
            if (mask & (1 << i)) {
                int pre = mask ^ (1 << i);
                if (dp[pre
                     ] == 1 && (g[i] & bitset<N>(pre)) == pre)
                    dp[mask] = 1; // i 有連到所有 pre
            }
        }
    }
    for (int
          mask = 0; mask < 1 << n; mask++) // 然後枚舉子集 dp
        for (int sub = mask; sub; --sub &= mask)
            dp[mask] = min(dp[mask], dp[sub] + dp[mask ^ sub]);
    cout << dp[(1 << n) - 1] << "\n";
}
```

9.5 Projects [8aa468]

```
void projects() { // 排程有權重問題，輸出價值最多且時間最少
    struct E { int from, to, w, id; };
    int n; cin >> n; vector<E> a(n + 1);
    for (int i = 1; i <= n; i++) {
        int u, v, w; cin >> u >> v >> w;
        a[i] = {u, v, w, i};
    }
    vector<array<ll, 2>> dp(n + 1); // w, time
    vector<array<int, 2>> rec(n + 1); // 有沒選，上個是誰
    sort(a.begin(), a.end());
    for (int i = 1; i <= n; i++) {
        int id = prev(
             lower_bound(all(a), {0, a[i].from}, [](E x, E y) {
            return x.to < y.to;
        })) - a.begin();
        dp[i] = dp[i - 1];
        ll nw = dp[id][0] + a[i].w;
        ll nt = dp[id][1] + a[i].to - a[i].from;
        if (dp[i][0] < nw || dp[i][0] == nw && dp[i][1] > nt) {
            dp[i] = {nw, nt};
            rec[i] = {1, id};
        }
    }
    vector<int> ans;
    for (int i = n; i != 0;) {
        if (rec[i][0]) {
            ans.push_back(a[i].id);
            i = rec[i][1];
        } else i--;
    }
}
```

9.6 Monotonic Queue [c9ba14]

```
// 應用: dp(i) = h(i) + max(A(j)), for l(i)≤j≤r(i)
// A(j) 可能包含 dp(j), h(i) 可 O(1)
void boundedKnapsack() {
    int n, k; // O(nk)
    vector<int> w(n), v(n), num(n);
    deque<int> q;
    // 於是我們將同餘的數分在同一組
    // 每次取出連續 num[i] 格中最大值
    // g_x = max(_{k=0}^num[i] (g'_{x-k} + v_i*k))
    // G_x = g'_{x} - v_i*x
    // x 代 x-k => v_i*(x-k)
    // g_x = max(_{k=0}^num[i] (G_{x-k} + v_i*x))
    vector<vector<ll>> dp(2, vector<ll>(k + 1));
    for (int i = 0; i < n; i++) {
        for (int r = 0; r < w[i]; r++) { // 餘數
            q.clear(); // q 記錄在 x = i 時的 dp 有單調性
            for (int x = 0; x * w[i] + r <= k; x++) {
                while (!q.empty() && q.front()
                      < x - num[i]) q.pop_front(); // 維護遞減
                ll nxt = dp[0][x * w[i] + r] - x * v[i];
                while (!q.empty() && dp[0][q.back() * w[i] +
                      r] - q.back() * v[i] < nxt) q.pop_back();
                q.push_back(x);
                dp[1][x * w[i] + r] = dp[0][q.front()
                     * w[i] + r] - q.front() * v[i] + x * v[i];
            }
        }
        swap(dp[0], dp[1]);
    }
    cout << dp[0][k] << "\n";
}
```

9.7 SOS [be203d]

```
// 使用情況: 跟 bit 與(被)包含有關, 且 x 在 1E6 左右
// 題目: 一數組, 問有多少所有數 & 起來為 0 的集合數
// dp[
     x] 代表包含 x 的 y 個數(比 x 大且 bit 1 全包含 x 的有幾個)
// 答案應該包含在 dp[0] 內，但是有重複元素，所以考慮容斥
// => ans = \sum _{i=0}^{n} (-1)^{pop_count(i)} 2^{dp[i]-1}
// => 全
     部為 0 的個數 - 至少一個為 1 的個數 + 至少兩個為 1 的個數
void solve() {
    int n; cin >> n; Z ans = 0;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];
    int m = __lg(*max_element(a.begin(), a.end())) + 1;
    // 定義 dp[mask] 為 mask 被包含於 a[i] 的 i 個數
    vector<ll> dp(1 << m);
    for (int i = 0; i < n; i++) dp[a[i]]++;
    for (int i = 0; i < m; i++) {
        for (int mask = 0; mask < 1 << m; mask++) {
            if (mask >> i & 1) {
                int pre = mask ^ (1 << i);
                dp[pre] += dp[mask];
            }
        }
    }
    for (int mask = 0; mask < 1 << m; mask++) {
        int sgn = __builtin_popcount(mask) & 1 ? -1 : 1;
        ans += sgn * (power(Z(2), dp[mask]) - 1);
    }
}
// x | y = x, 代表包含於 x 的 y 個數, 定義為 dp[x][0]
// x & y = x, 代表包含 x 的 y 個數, 定義為 dp[x][1]
// x & y
      != 0, 代表至少有一個位元都為 1 的 y 個數, = n - dp[~x][0]
void solve() {
    int n; cin >> n;
    vector<int> a(n);
    map<int, int> mp;
    for (int i = 0; i < n; i++) {
        cin >> a[i]; mp[a[i]]++;
    }
    int m = __lg(*max_element(a.begin(), a.end())) + 1;
    vector<array<ll, 2>> dp(1 << m);
    for (int i = 0; i < n; i++) dp[a[i]][0]++, dp[a[i]][1]++;
    for (int i = 0; i < m; i++) {
        for (int mask = 0; mask < 1 << m; mask++) {
            if (mask >> i & 1) {
                int pre = mask ^ (1 << i);
                dp[mask][0] += dp[pre][0];
                dp[pre][1] += dp[mask][1];
            }
        }
    }
    for (int i = 0; i < n; i++) {
        cout << dp[a[i]][0] << " " << dp[a[i]][1] <<
            " " << n - (dp[((1 << m) - 1) ^ a[i]][0]) << "\n";
    }
}
```

9.8 CHT [ce439f]

```
// 應用: dp(x) = C(x) + min/max(A(i) * x + B(i)), for i < x
struct Line { // x 盡量從 1 開始
    ll m, b;
    Line(ll m = 0, ll b = 0) : m(m), b(b) {}
    ll eval(ll x) { return m * x + b; }
};
struct CHT { // 斜率單調
    int lptr = 0, rptr = 0;
    vector<Line> hull;
    CHT(Line init = Line()) { hull.push_back(init); }
    bool frontBad(Line &l1, Line &l2, ll x) {
        // 斜率遞減、查詢遞增，因此只要左直線的 Y >= 右直線的 Y
        // 代表查詢的當下，右線段的高度已經低於左線段了
        return l1.eval(x) >= l2.eval(x);
    }
    bool backBad(Line &l1, Line &l2, Line &l3) {
        // 斜率遞減、上凸包、取 min
        // 因此只要 l2 跟
            l3 的 X 交點 <= l1 跟 l3 的 X 交點，l2 就用不到了
        return (l3.b - l2.b)
            * (l1.m - l3.m) <= (l3.b - l1.b) * (l2.m - l3.m);
    }
    void addLine(Line l) {
        while (rptr - lptr > 0 && backBad(hull[rptr
            - 1], hull[rptr], l)) hull.pop_back(), rptr--;
        hull.push_back(l), rptr++;
    }
    ll query(ll x) { // 查詢沒單調性需要二分搜
        while (rptr - lptr > 0 &&
            frontBad(hull[lptr], hull[lptr + 1], x)) lptr++;
        return hull[lptr].eval(x);
    }
};
```

9.9 DNC [9fea10]

```
// 應用: 切 k 段問題, 且滿足四邊形不等式
// w(a,c) + w(b,d) ≤(≥) w(a,d) + w(b,c)
// dp[k][j] = min(dp[k - 1][i] + cost[i][j])
// cost: (i, j]
constexpr int N = 3E3 + 5;
constexpr ll inf = 4E18;
ll dp[N][N]; // 1-based
ll getCost(int l, int r) {}
void rec(int k, int l, int r, int optl, int optr) {
    if (l > r) return;
    int m = (l + r) >> 1, opt = -1;
    dp[k][m] = inf;
    for (int i = max(k, optl); i <= min(m, optr); i++) {
        // 注意 i 的範圍、 get_cost 與 dp 的邊界
        ll cur = dp[k - 1][i] + getCost(i, m);
        if (cur < dp[k][m]) dp[k][m] = cur, opt = i;
    }
    rec(k, l, m - 1, optl, opt);
    rec(k, m + 1, r, opt, optr);
}
void DNC() {
    // first build cost...
    for (int i = 1; i <= n; i++) dp[1][i] = getCost(1, i);
    for (int i = 2; i <= k; i++) rec(i, 1, n, 1, n);
    cout << dp[k][n] << "\n";
}
```

9.10 LiChao Segment Tree [2a9325]

```
// 應用: dp(i) = h(i) + min/max(A(j)X(i) + B(j)), for j ≤ r(i)
//          y    = c   +         m  x   +   b
template<class T, class F = less<ll>>
struct LiChaoSeg {
    F cmp = F();
    static const T inf = max(numeric_limits
         <T>::lowest() / 2, numeric_limits<T>::max() / 2, F());
    struct Line {
        T m, b;
        Line(T m = 0, T b = inf) : m(m), b(b) {}
        T eval(T x) const { return m * x + b; }
    };
    struct Node {
        Line line;
        ll l = -1, r = -1;
    };
    ll n;
    vector<Node> nd;
    LiChaoSeg(ll n) : n(n) { newNode(); }
    void addLine(Line line) { update(0, 0, n, line); }
    void rangeAddLine(Line line,
         ll ql, ll qr) { rangeUpdate(0, 0, n, ql, qr, line); }
    T query(ll x) { return query(x, 0, 0, n); }
private:
    int newNode() {
        nd.emplace_back();
        return nd.size() - 1;
    }
    void update(int p, ll l, ll r, Line line) {
        ll m = (l + r) / 2;
        bool left = cmp(line.eval(l), nd[p].line.eval(l));
        bool mid = cmp(line.eval(m), nd[p].line.eval(m));
        if (mid) swap(nd[p].line, line);
        if (r - l == 1) return;
        if (left != mid) {
            if (nd[p].l == -1) nd[p].l = newNode();
            update(nd[p].l, l, m, line);
        } else {
            if (nd[p].r == -1) nd[p].r = newNode();
            update(nd[p].r, m, r, line);
        }
    }
    void rangeUpdate
     (int p, ll l, ll r, ll ql, ll qr, Line line) {
        if (r <= ql || l >= qr) return;
        if (ql <= l && r <= qr) return update(p, l, r, line);
        if (nd[p].l == -1) nd[p].l = newNode();
        if (nd[p].r == -1) nd[p].r = newNode();
        ll m = (l + r) / 2;
        rangeUpdate(nd[p].l, l, m, ql, qr, line);
        rangeUpdate(nd[p].r, m, r, ql, qr, line);
    }
    T query(ll x, int p, ll l, ll r) {
        if (p == -1) return inf;
        ll m = (l + r) / 2;
        if (x < m) return min(
            nd[p].line.eval(x), query(x, nd[p].l, l, m), cmp);
        else return min(
            nd[p].line.eval(x), query(x, nd[p].r, m, r), cmp);
    }
};
```

10 Geometry

10.1 Basic [d41d8c]

```
const double eps = 1E-9;
template<class T>
struct Pt {
    T x, y;
    Pt(T x = 0, T y = 0) : x(x), y(y) {}
    Pt operator-() const { return Pt(-x, -y); }
    Pt operator+(Pt p) const { return Pt(x + p.x, y + p.y); }
    Pt operator-(Pt p) const { return Pt(x - p.x, y - p.y); }
    Pt operator*(T k) const { return Pt(x * k, y * k); }
    Pt operator/(T k) const { return Pt(x / k, y / k); }
    bool operator
        ==(Pt p) const { return x == p.x && y == p.y; }
    bool operator
        !=(Pt p) const { return x != p.x || y != p.y; }
    friend istream &operator>>(istream &is, Pt &p) {
        return is >> p.x >> p.y;
    }
    friend ostream &operator<<(ostream &os, const Pt &p) {
        return os << "(" << p.x << ", " << p.y << ")";
    }
};
int sign(double x)
{ return fabs(x) <= eps ? 0 : (x > 0 ? 1 : -1); }
using P = Pt<double>;

struct Line { P a, b; };
double dot(P a, P b) { return a.x * b.x + a.y * b.y; }
double cross(P a, P b) { return a.x * b.y - a.y * b.x; }
double square(P p) { return dot(p, p); }
double abs(P p) { return sqrt(square(p)); }
double dist(P a, P b) { return abs(a - b); }
double abs(Line l) { return abs(l.a - l.b); }
int dir(P p, Line l) // left -1, right 1, on 0
{ return -sign(cross(l.b - l.a, p - l.a)); }
bool btw(P p, Line l) // c on segment ab?
{ return dir(p, l) == 0 && sign(dot(p - l.a, p - l.b)) <= 0; }
P norm(P p) { return p / abs(p); }
P rot(P p) { return { -p.y, p.x }; } // 90 degree CCW
P rot(P p, double d) {
    double c = cos(d), s = sin(d);
    return { p.x * c - p.y * s, p.x * s + p.y * c };
```

```
}

bool parallel(Line l1, Line l2)
{ return cross(l1.b - l1.a, l2.b - l2.a) == 0; }
P lineIntersection(Line l1, Line l2)
{ return l1.a + (l1.b - l1.a) * (cross(l2.b
     - l2.a, l1.a - l2.a) / cross(l2.b - l2.a, l1.a - l1.b)); }
bool pointOnSegment(P p, Line l)
{ return dir(p, l) == 0 && sign(dot(p - l.a, p - l.b)) <= 0; }
P projvec(P p, Line l) {
    P v = l.b - l.a;
    return l.a + v * (dot(p - l.a, v) / square(v));
}
// 0 : not intersect
// 1 : strictly intersect
// 2 : overlap
// 3 : intersect at endpoint
tuple<int, P, P> segmentIntersection(Line l1, Line l2) {
    if (max(l1.a.x, l1.b.x) < min(l2.a.x, l2.b.x) ||
        min(l1.a.x, l1.b.x) > max(l2.a.x, l2.b.x) ||
        max(l1.a.y, l1.b.y) < min(l2.a.y, l2.b.y) ||
        min(l1.a.y, l1.b.y) > max(l2.a.y, l2.b.y))
        return {0, {}, {}};
    if (cross(l1.b - l1.a, l2.b - l2.a) == 0) {
        if (cross(l1.b - l1.a, l2.a - l1.a) != 0) {
            return {0, {}, {}};
        } else {
            auto maxx1 = max(l1.a.x, l1.b.x);
            auto minx1 = min(l1.a.x, l1.b.x);
            auto maxy1 = max(l1.a.y, l1.b.y);
            auto miny1 = min(l1.a.y, l1.b.y);
            auto maxx2 = max(l2.a.x, l2.b.x);
            auto minx2 = min(l2.a.x, l2.b.x);
            auto maxy2 = max(l2.a.y, l2.b.y);
            auto miny2 = min(l2.a.y, l2.b.y);
            P p1(max(minx1, minx2), max(miny1, miny2));
            P p2(min(maxx1, maxx2), min(maxy1, maxy2));
            if (!pointOnSegment(p1, l1)) swap(p1.y, p2.y);
            if (p1 == p2) return {3, p1, p2};
            else return {2, p1, p2};
        }
    }
    auto cp1 = cross(l2.a - l1.a, l2.b - l1.a);
    auto cp2 = cross(l2.a - l1.b, l2.b - l1.b);
    auto cp3 = cross(l1.a - l2.a, l1.b - l2.a);
    auto cp4 = cross(l1.a - l2.b, l1.b - l2.b);
    if ((cp1 > 0
         && cp2 > 0) || (cp1 < 0 && cp2 < 0) || (cp3 > 0 && cp4
         > 0) || (cp3 < 0 && cp4 < 0)) return {0, P(), P()};
    P p = lineIntersection(l1, l2);
    if (cp1 != 0
         && cp2 != 0 && cp3 != 0 && cp4 != 0) return {1, p, p};
    else return {3, p, p};
}

vector<P> convexHull(vector<P> a) {
    sort(a.begin(), a.end(), [](const P &l, const P &r) {
        return l.x == r.x ? l.y < r.y : l.x < r.x;
    });
    a.resize(unique(a.begin(), a.end()) - a.begin());
    if (a.size() <= 1) return a;
    vector<P> h(a.size() + 1);
    int s = 0, t = 0;
    for (int i = 0; i < 2; i++, s = --t) {
        for (P p : a) {
            while (t >= s + 2 && cross
                 (h[t - 1] - h[t - 2], p - h[t - 2]) <= 0) t--;
            h[t++] = p;
        }
        reverse(a.begin(), a.end());
    }
    return {h.begin(), h.begin() + t};
}

double distPL(P &p, Line &l)
{ return abs(cross(l.a - l.b, l.a - p)) / abs(l); }
double distancePS(P &p, Line &l) {
    if (dot(p - l.a, l.b - l.a) < 0) return dist(p, l.a);
    if (dot(p - l.b, l.a - l.b) < 0) return dist(p, l.b);
    return distPL(p, l);
}
double distanceSS(Line l1, Line l2) {
    if (get<0>(segmentIntersection(l1, l2)) != 0) return 0.0;
    return min({distancePS(l1.a, l2), distancePS(l1
         .b, l2), distancePS(l2.a, l1), distancePS(l2.b, l1)});
}

bool lineIntersectsPolygon(Line l, const vector<P> &p) {
    int n = p.size();
    P a = l.a, b = l.b;
    for (int i = 0; i < n; i++) {
        Line seg {p[i], p[(i + 1) % n]};
        if (cross(b - a, seg.a - a)
             == 0 || cross(b - a, seg.b - a) == 0) return true;
        if ((cross(b - a, seg.a - a)
             > 0) ^ (cross(b - a, seg.b - a) > 0)) return true;
    }
    return false;
}
bool pointInPolygon(P a, const vector<P> &p) {
    int n = p.size(), t = 0;
    for (int i = 0; i < n; i++)
        if (pointOnSegment
             (a, {p[i], p[(i + 1) % n]})) return true;
    for (int i = 0; i < n; i++) {
        P u = p[i], v = p[(i + 1) % n];
        if (u.x
             < a.x && v.x >= a.x && dir(a, {v, u}) < 0) t ^= 1;
        if (u.x
             >= a.x && v.x < a.x && dir(a, {u, v}) < 0) t ^= 1;
    }
    return t == 1;
}
// 0 : strictly outside
// 1 : on boundary
// 2 : strictly inside
int pointInConvexPolygon(P a, const vector<P> &p) {
    int n = p.size();
    if (n == 0) return 0;
    else if
         (n <= 2) return pointOnSegment(a, {p[0], p.back()});
    if (pointOnSegment(a, {p[0], p[1]})
         || pointOnSegment(a, {p[0], p[n - 1]})) return 1;
    else if (dir(a, {p[0],
         p[1]}) < 0 || dir(a, {p[0], p[n - 1]}) < 0) return 0;
    int lo = 1, hi = n - 2;
    while (lo < hi) {
        int x = (lo + hi + 1) / 2;
        if (dir(a, {p[0], p[x]}) < 0) lo = x;
        else hi = x - 1;
    }
    if (dir(a, {p[lo], p[lo + 1]}) < 0) return 2;
    else return pointOnSegment(a, {p[lo], p[lo + 1]});
}
bool segmentInPolygon(Line l, const vector<P> &p) {
    int n = p.size();
    if (!pointInPolygon(l.a, p)) return false;
    if (!pointInPolygon(l.b, p)) return false;
    for (int i = 0; i < n; i++) {
        auto u = p[i];
        auto v = p[(i + 1) % n];
        auto w = p[(i + 2) % n];
        auto [t, p1, p2] = segmentIntersection(l, {u, v});
        if (t == 1) return false;
        if (t == 0) continue;
        if (t == 2) {
            if (pointOnSegment(v, l) && v != l.a && v !=
                 l.b && cross(u - v, w - v) < 0) return false;
        } else {
            if (p1 != u && p1 != v) {
                if (dir(l.a, {v, u})
                     < 0 || dir(l.b, {v, u}) < 0) return false;
            } else if (p1 == v) {
                if (l.a == v) {
                    if (dir(u, l) < 0) {
                        if (dir(w, l) < 0 &&
                             dir(w, {u, v}) < 0) return false;
                    } else if (dir(w, l) <
                         0 || dir(w, {u, v}) < 0) return false;
                } else if (l.b == v) {
                    if (dir(u, {l.b, l.a}) < 0) {
                        if (dir(w, {l.b, l.a}) < 0 &&
                             dir(w, {u, v}) < 0) return false;
                    } else if (dir(w, {l.b, l.a}) <
                         0 || dir(w, {u, v}) < 0) return false;
                } else {
                    if (dir(u, l) < 0) {
                        if (dir(w, {l.b, l.a}) < 0 ||
                             dir(w, {u, v}) < 0) return false;
                    } else if (dir(w, l) <
                         0 || dir(w, {u, v}) < 0) return false;
                }
            }
        }
    }
    return true;
}
vector<P> hp(vector<Line> lines) {
    auto sgn = [](P p)
         { return p.y > 0 || (p.y == 0 && p.x > 0) ? 1 : -1; };
    sort(lines.begin(), lines.end(), [&](auto l1, auto l2) {
        auto d1 = l1.b - l1.a;
        auto d2 = l2.b - l2.a;
        if (sgn(d1) != sgn(d2))
            return sgn(d1) == 1;
        return cross(d1, d2) > 0;
    });
    deque<Line> ls;
    deque<P> ps;
    for (auto l : lines) {
        if (ls.empty()) {
            ls.push_back(l);
            continue;
        }
        while (!ps.empty() && dir
             (ps.back(), l) >= 0) ps.pop_back(), ls.pop_back();
        while (!ps.empty() && dir
             (ps[0], l) >= 0) ps.pop_front(), ls.pop_front();
        if (cross(l.b - l.a, ls.back().b - ls.back().a) == 0) {
            if (dot
                 (l.b - l.a, ls.back().b - ls.back().a) > 0) {
                if (dir(ls.back().a, l) >= 0) {
                    assert(ls.size() == 1);
                    ls[0] = l;
                }
                continue;
            }
            return {};
        }
        ps.push_back(lineIntersection(ls.back(), l));
        ls.push_back(l);
    }
    while (!ps.empty() && dir(ps.back(), ls[0]) >= 0)
        ps.pop_back(), ls.pop_back();
    if (ls.size() <= 2) return {};
    ps.push_back(lineIntersection(ls[0], ls.back()));
    return vector(ps.begin(), ps.end());
}
```

10.2 Min Euclidean Distance [cfb429]

```
// recursive solution
void minEuclideanDistance() {
    int n; cin >> n;
    const ll inf = 8E18;
    vector<P> a(n);
    for (int i = 0; i < n; i++) {
        ll x, y; cin >> x >> y;
        a[i] = P(x, y);
    }
    struct sortY { bool operator()(
         const P &a, const P &b) const { return a.y < b.y; } };
    struct sortXY {
        bool operator()(const P &a, const P &b) const {
            return a.x == b.x ? a.y < b.y : a.x < b.x;
        }
    };
    sort(a.begin(), a.end(), sortXY());
    vector<P> t(n);
    auto divide = [&](auto &&self, int l, int r) -> ll {
        if (l == r) return inf;
        int m = (l + r) / 2;
        ll ans = min(self(self, l, m), self(self, m + 1, r));
        ll midval = a[m].x;
        ll p = 0;
        for (int i = l; i <= r; i++)
            if ((midval - a[i].x) * (midval - a[i].x) <= ans)
                t[p++] = a[i];
        sort(t.begin(), t.begin() + p, sortY());
        for (int i = 0; i < p; i++) {
            for (int j = i + 1; j < p; j++) {
                ans = min(ans, square(t[i] - t[j]));
                if ((t[i].y -
                      t[j].y) * (t[i].y - t[j].y) > ans) break;
            }
        }
        return ans;
    };
    cout << divide(divide, 0, n - 1) << "\n";
}
// K-D tree solution
struct Info {
    static constexpr int DIM = 2;
    array<ll, DIM> x, L, R;
    ll distl, distr;
    ll f(const Info &i) {
        ll ret = 0;
        if (i.L[0]
              > x[0]) ret += (i.L[0] - x[0]) * (i.L[0] - x[0]);
        if (i.R[0]
              < x[0]) ret += (x[0] - i.R[0]) * (x[0] - i.R[0]);
        if (i.L[1]
              > x[1]) ret += (i.L[1] - x[1]) * (i.L[1] - x[1]);
        if (i.R[1]
              < x[1]) ret += (x[1] - i.R[1]) * (x[1] - i.R[1]);
        return ret;
    }
    void pull(const Info &l, const Info &r) {
        distl = f(l), distr = f(r);
    }
};
struct KDTree { // 1-indexed
    static constexpr int DIM = Info::DIM;
    int n, rt;
    vector<Info> info;
    vector<int> l, r;
    KDTree(const vector<Info> &info
         ) : n(info.size()), info(info), l(n + 1), r(n + 1) {
        rt = build(1, n);
    }
    void pull(int p) {
        info[p].L = info[p].R = info[p].x;
        info[p].pull(info[l[p]], info[r[p]]);
        for (int ch : {l[p], r[p]}) {
            if (!ch) continue;
            for (int k = 0; k < DIM; k++) {
                info[p
                     ].L[k] = min(info[p].L[k], info[ch].L[k]);
                info[p
                     ].R[k] = max(info[p].R[k], info[ch].R[k]);
            }
        }
    }
    int build(int l, int r) {
        if (r == l) return 0;
        int m = (l + r) / 2;
        array<double, DIM> av = {}, va = {};
        for (int i = l; i < r; i++)
            for (int d = 0; d < DIM; d++)
                av[d] += info[i].x[d];
        for (int d = 0; d < DIM; d++)
            av[d] /= (double)(r - l);
        for (int i = l; i < r; i++)
            for (int d = 0; d < DIM; d++)
                va[d] += (info[
                     i].x[d] - av[d]) * (info[i].x[d] - av[d]);
        int dep
              = max_element(va.begin(), va.end()) - va.begin();
        nth_element(info
             .begin() + l, info.begin() + m, info.begin() + r,
            [&](const Info &x, const
                  Info &y) { return x.x[dep] < y.x[dep]; });
        this->l[m] = build(l, m);
        this->r[m] = build(m + 1, r);
        pull(m); return m;
    }
    ll ans = 9E18;
    ll dist(int a, int b) {
        return (info[a].x[0]
              - info[b].x[0]) * (info[a].x[0] - info[b].x[0]) +
        (info[a].x[1]
              - info[b].x[1]) * (info[a].x[1] - info[b].x[1]);
    }
    void query(int p, int x) {
        if (!p) return;
        if (p != x) ans = min(ans, dist(x, p));
        ll distl = info[x].f(info[l[p]]);
        ll distr = info[x].f(info[r[p]]);
        if (distl < ans && distr < ans) {
            if (distl < distr) {
                query(l[p], x);
                if (distr < ans) query(r[p], x);
            } else {
                query(r[p], x);
                if (distl < ans) query(l[p], x);
            }
        } else {
            if (distl < ans) query(l[p], x);
            if (distr < ans) query(r[p], x);
        }
    }
};
```

10.3 Max Euclidean Distance [4e338a]

```
tuple<ll, int, int> maxEuclideanDistance(vector<P> a) {
    auto get = [&](P p, Line l) -> ll {
        return abs(cross(l.a - l.b, l.a - p));
    };
    ll res = 0; int n = a.size(), x, y, id = 2;
    a.push_back(a.front());
    if (n <= 2) return {abs2(a[0] - a[1]), 0, 1};
    for (int i = 0; i < n; i++) {
        while (get(a[id], {a[i], a[i
              + 1]}) <= get(a[(id + 1) % n], {a[i], a[i + 1]}))
            id = (id + 1) % n;
        if (res < abs2(a[i] - a[id])) {
            res = abs2(a[i] - a[id]);
            x = i, y = id;
        }
        if (res < abs2(a[i + 1] - a[id])) {
            res = abs2(a[i + 1] - a[id]);
            x = i + 1, y = id;
        }
    }
    return {res, x, y};
}
```

10.4 Lattice Points [2e0d5a]

```
void latticePoints() {
    // Area 求法與 Polygon 內整數點數
    int n; cin >> n;
    vector<P> polygon(n);
    for (int i = 0; i < n; i++) cin >> polygon[i];
    ll area = 0;
    for (int i = 0; i < n; i++)
        area += cross(polygon[i], polygon[(i + 1) % n]);
    area = abs(area);
    auto countBoundaryPoints
          = [](const vector<P> &polygon) -> ll {
        ll res = 0;
        int n = polygon.size();
        for (int i = 0; i < n; i++) {
            ll dx = polygon[(i + 1) % n].x - polygon[i].x;
            ll dy = polygon[(i + 1) % n].y - polygon[i].y;
            res += __gcd(abs(dx), abs(dy));
        }
        return res;
    };
    ll res = countBoundaryPoints(polygon);
    ll ans = (area - res + 2) / 2;
    cout << ans << " " << res << "\n";
}
```

10.5 Min Circle Cover [71b50f]

```
pair<double, P> minCircleCover(vector<P> a) {
    shuffle(a.begin(), a.end(), rng);
    int n = a.size();
    P c = a[0]; double r = 0;
    for (int i = 1; i < n; i++) {
        if (sign(abs(c - a[i]) - r) > 0) {
            c = a[i], r = 0;
            for (int j = 0; j < i; j++) {
                if (sign(abs(c - a[j]) - r) > 0) {
                    c = (a[i] + a[j]) / 2.0;
                    r = abs(c - a[i]);
                    for (int k = 0; k < j; k++) {
                        if (sign(abs(c - a[k]) - r) > 0) {
                            P p = (a[j] + a[i]) / 2;
                            P q = (a[j] + a[k]) / 2;
                            if (cross(a[j] - a[i],
                                  a[k] - a[j]) == 0) continue;
                            c = lineIntersection
                                 ({p, p + rot(a[j] - a[i])
                                 }, {q, q + rot(a[k] - a[j])});
                            r = abs(c - a[i]);
                        }
                    }
                }
            }
        }
    }
    return {r, c};
}
```

10.6 Min Rectangle Cover [bde8e6]

```
pair<double, vector<P>> minRectangleCover(vector<P> p) {
    if (p.size() <= 2) return {0, {}};
    auto get = [&](P p, Line l) -> double {
        return abs(cross(l.a - l.b, l.a - p));
    }; // line 到 p 圍成的四邊形面積
    int n = p.size(), j = 2, l = 1, r = 1;
    p.push_back(p.front());
    double ans = 8E18;
    vector<P> ps;
    for (int i = 0; i < n; i++) {
        while (get(p[j], {p[i], p[
             i + 1]}) <= get(p[(j + 1) % n], {p[i], p[i + 1]}))
            j = (j + 1) % n;
        while (dot(p[i + 1] - p[i], p[r] - p[i
             ]) <= dot(p[i + 1] - p[i], p[(r + 1) % n] - p[i]))
            r = (r + 1) % n;
        if (i == 0) l = j;
        while (dot(p[i + 1] - p[i], p[l] - p[i
             ]) >= dot(p[i + 1] - p[i], p[(l + 1) % n] - p[i]))
            l = (l + 1) % n;
        double area = get(p[j], {p[i], p[i + 1]});
        double w = dot(p[i] - p[i + 1],
              p[l] - p[i]) + dot(p[i + 1] - p[i], p[r] - p[i]);
        area *= w / square(p[i + 1] - p[i]);
        if (area < ans) {
            ps.clear(), ans = area;
            Line l1 {p[i], p[i + 1]};
            for (auto u : {p[r], p[j], p[l], p[i]}) {
                if (u == l1.b) {
                    ps.push_back(u);
                    l1 = {u, u + rot(l1.b - l1.a)};
                } else {
                    Line l2 = {u, u + rot(l1.b - l1.a)};
                    P res = lineIntersection(l1, l2);
                    ps.push_back(res);
                    l1 = {res, u};
                }
            }
        }
    }
    return {ans, ps};
}
```

10.7 Polygon Union Area [dc0989]

```
double polygonUnion(vector<vector<P>> ps) { // CCW needed
    int n = ps.size();
    for (auto &v : ps) v.push_back(v[0]);
    double res = 0;
    auto seg = [&](P o, P a, P b) -> double {
        if (b.x - a.x == 0) return (o.y - a.y) / (b.y - a.y);
        return (o.x - a.x) / (b.x - a.x);
    };
    for (int pi = 0; pi < n; pi++) {
        for (int i = 0; i + 1 < ps[pi].size(); i++) {
            vector<pair<double, int>> e;
            e.emplace_back(0, 0);
            e.emplace_back(1, 0);
            for (int pj = 0; pj < n; pj++) {
                if (pi == pj) continue;
                for (int j = 0; j + 1 < ps[pj].size(); j++) {
                    auto c1 = cross(ps[pi][i + 1]
                          - ps[pi][i], ps[pj][j] - ps[pi][i]);
                    auto c2 = cross(ps[pi][i + 1] -
                         ps[pi][i], ps[pj][j + 1] - ps[pi][i]);
                    if (c1 == 0 && c2 == 0) {
                        if (dot(ps[pi][i
                             + 1] - ps[pi][i], ps[pj][j + 1] -
                             ps[pj][j]) > 0 && (pi - pj) > 0) {
                            e.emplace_back(seg(ps[pj][j],
                                 ps[pi][i], ps[pi][i + 1]), 1);
                            e.emplace_back
                                 (seg(ps[pj][j + 1], ps
                                 [pi][i], ps[pi][i + 1]), -1);
                        }
                    } else {
                        auto s1 = cross(ps[pj][j + 1] -
                             ps[pj][j], ps[pi][i] - ps[pj][j]);
                        auto s2 = cross(ps[pj][j + 1] - ps[pj
                             ][j], ps[pi][i + 1] - ps[pj][j]);
                        if (c1 >= 0 && c2 < 0)
                             e.emplace_back(s1 / (s1 - s2), 1);
                        else if (c1 < 0 && c2 >= 0) e
                             .emplace_back(s1 / (s1 - s2), -1);
                    }
                }
            }
            sort(e.begin(), e.end());
            double pre = clamp(e[0].first, 0.0, 1.0), sum = 0;
            int cov = e[0].second;
            for (int j = 1; j < e.size(); j++) {
                double now = clamp(e[j].first, 0.0, 1.0);
                if (!cov) sum += now - pre;
                cov += e[j].second;
                pre = now;
            }
            res += cross(ps[pi][i], ps[pi][i + 1]) * sum;
        }
    }
    return res / 2;
}
```

11 Polynomial

11.1 FFT [e258ad]

```
const double PI = acos(-1.0);
using cd = complex<double>;
vector<int> rev;
void fft(vector<cd> &a, bool inv) {
    int n = a.size();
    if (int(rev.size()) != n) {
        int k = __builtin_ctz(n) - 1;
        rev.resize(n);
        for (int i = 0; i < n; i++)
            rev[i] = rev[i >> 1] >> 1 | (i & 1) << k;
    }
    for (int i = 0; i < n; i++)
        if (rev[i] < i) swap(a[i], a[rev[i]]);
    for (int k = 1; k < n; k *= 2) {
        double ang = (inv ? -1 : 1) * PI / k;
        cd wn(cos(ang), sin(ang));
        for (int i = 0; i < n; i += 2 * k) {
            cd w(1);
            for (int j = 0; j < k; j++, w = w * wn) {
                cd u = a[i + j];
                cd v = a[i + j + k] * w;
                a[i + j] = u + v;
                a[i + j + k] = u - v;
            }
        }
    }
    if (inv) for (auto &x : a) x /= n;
}
template<class T>
vector<T> Multiple(const vector<T> &a, const vector<T> &b) {
    vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.end());
    int n = 1, tot = a.size() + b.size() - 1;
    while (n < tot) n *= 2;
    fa.resize(n), fb.resize(n);
    fft(fa, false), fft(fb, false);
    for (int i = 0; i < n; i++)
        fa[i] = fa[i] * fb[i];
    fft(fa, true);
    vector<T> res(tot);
    for (int i = 0; i < tot; i++)
        res[i] = fa[i].real(); // use llround if need
    return res;
}
```

11.2 NTT [6caf78]

```
template<int V, int P>
Mint<P> CInv = Mint<P>(V).inv();
vector<int> rev;
template<int P>
vector<Mint<P>> roots {0, 1};
template<int P>
Mint<P> findPrimitiveRoot() {
    Mint<P> i = 2;
    int k = __builtin_ctz(P - 1);
    while (true) {
        if (power(i, (P - 1) / 2) != 1) break;
        i += 1;
    }
    return power(i, (P - 1) >> k);
}
template<int P>
Mint<P> primitiveRoot = findPrimitiveRoot<P>();
template<>
Mint<998244353> primitiveRoot<998244353> {31};
```

```
template<int P>
void dft(vector<Mint<P>> &a) {
    int n = a.size();
    if (int(rev.size()) != n) {
        int k = __builtin_ctz(n) - 1;
        rev.resize(n);
        for (int i = 0; i < n; i++)
            rev[i] = rev[i >> 1] >> 1 | (i & 1) << k;
    }
    for (int i = 0; i < n; i++)
        if (rev[i] < i) swap(a[i], a[rev[i]]);
    if (roots<P>.size() < n) {
        int k = __builtin_ctz(roots<P>.size());
        roots<P>.resize(n);
        while ((1 << k) < n) {
            auto e = power(primitiveRoot
                 <P>, 1 << (__builtin_ctz(P - 1) - k - 1));
            for (int i = 1 << (k - 1); i < (1 << k); i++) {
                roots<P>[2 * i] = roots<P>[i];
                roots<P>[2 * i + 1] = roots<P>[i] * e;
            }
            k++;
        }
    }
    for (int k = 1; k < n; k *= 2) {
        for (int i = 0; i < n; i += 2 * k) {
            for (int j = 0; j < k; j++) {
                Mint<P> u = a[i + j];
                Mint<P> v = a[i + j + k] * roots<P>[k + j];
                a[i + j] = u + v;
                a[i + j + k] = u - v;
            }
        }
    }
}
template<int P>
void idft(vector<Mint<P>> &a) {
    int n = a.size();
    reverse(a.begin() + 1, a.end());
    dft(a);
    Mint<P> inv = (1 - P) / n;
    for (int i = 0; i < n; i++) a[i] *= inv;
}
template<int P = 998244353>
struct Poly : public vector<Mint<P>> {
    using Value = Mint<P>;
    Poly() : vector<Value>() {}
    explicit Poly(int n) : vector<Value>(n) {}
    explicit Poly(const vector<Value> &a) : vector<Value>(a) {}
    Poly(const
          initializer_list<Value> &a) : vector<Value>(a) {}
    template<class InputIt, class = _RequireInputIter<InputIt>>
    explicit Poly(InputIt
          first, InputIt last) : vector<Value>(first, last) {}
    template<class F>
    explicit Poly(int n, F f) : vector<Value>(n) {
        for (int i = 0; i < n; i++)
            (*this)[i] = f(i);
    }
    Poly shift(int k) const {
        if (k >= 0) {
            auto b = *this;
            b.insert(b.begin(), k, 0);
            return b;
        } else if (this->size() <= -k) {
            return Poly();
        } else {
            return Poly(this->begin() + (-k), this->end());
        }
    }
    Poly trunc(int k) const {
        Poly f = *this;
        f.resize(k);
        return f;
    }
    friend Poly operator+(const Poly &a, const Poly &b) {
        Poly res(max(a.size(), b.size()));
        for (int i = 0; i < a.size(); i++)
            res[i] += a[i];
        for (int i = 0; i < b.size(); i++)
            res[i] += b[i];
        return res;
    }
    friend Poly operator-(const Poly &a, const Poly &b) {
        Poly res(max(a.size(), b.size()));
        for (int i = 0; i < a.size(); i++)
            res[i] += a[i];
        for (int i = 0; i < b.size(); i++)
            res[i] -= b[i];
        return res;
    }
    friend Poly operator-(const Poly &a) {
        vector<Value> res(a.size());
        for (int i = 0; i < int(res.size()); i++)
            res[i] = -a[i];
        return Poly(res);
    }
    friend Poly operator*(Poly a, Poly b) {
        if (a.size() == 0 || b.size() == 0)
            return Poly();
        if (a.size() < b.size()) swap(a, b);
        int n = 1, tot = a.size() + b.size() - 1;
        while (n < tot) n *= 2;
        if (((P - 1) & (n - 1)) != 0 || b.size() < 128) {
            Poly c(a.size() + b.size() - 1);
            for (int i = 0; i < a.size(); i++)
                for (int j = 0; j < b.size(); j++)
                    c[i + j] += a[i] * b[j];
            return c;
        }
        a.resize(n), b.resize(n);
        dft(a), dft(b);
        for (int i = 0; i < n; i++)
            a[i] *= b[i];
        idft(a);
        a.resize(tot);
        return a;
    }
    friend Poly operator*(Value a, Poly b) {
        for (int i = 0; i < int(b.size()); i++)
            b[i] *= a;
        return b;
    }
    friend Poly operator*(Poly a, Value b) {
        for (int i = 0; i < int(a.size()); i++)
            a[i] *= b;
        return a;
    }
    friend Poly operator/(Poly a, Value b) {
        for (int i = 0; i < int(a.size()); i++)
            a[i] /= b;
        return a;
    }
    Poly &operator+=(Poly b) {
        return (*this) = (*this) + b;
    }
    Poly &operator-=(Poly b) {
        return (*this) = (*this) - b;
    }
    Poly &operator*=(Poly b) {
        return (*this) = (*this) * b;
    }
    Poly &operator*=(Value b) {
        return (*this) = (*this) * b;
    }
    Poly &operator/=(Value b) {
        return (*this) = (*this) / b;
    }
    Poly deriv() const {
        if (this->empty()) return Poly();
        Poly res(this->size() - 1);
        for (int i = 0; i < this->size() - 1; i++)
            res[i] = (i + 1) * (*this)[i + 1];
        return res;
    }
    Poly integr() const {
        Poly res(this->size() + 1);
        for (int i = 0; i < this->size(); i++)
            res[i + 1] = (*this)[i] / (i + 1);
        return res;
    }
    Poly inv(int m) const {
        Poly x{(*this)[0].inv()};
        int k = 1;
        while (k < m) {
            k *= 2;
            x = (x * (Poly{2} - trunc(k) * x)).trunc(k);
        }
        return x.trunc(m);
    }
    Poly log(int m) const {
        return (deriv() * inv(m)).integr().trunc(m);
    }
    Poly exp(int m) const {
        Poly x{1};
        int k = 1;
        while (k < m) {
            k *= 2;
            x = (x * (Poly{1} - x.log(k) + trunc(k))).trunc(k);
        }
        return x.trunc(m);
    }
    Poly pow(int k, int m) const {
        int i = 0;
        while (i < this->size() && (*this)[i] == 0) i++;
        if (i == this->size() || 1LL * i * k >= m)
            return Poly(m);
        Value v = (*this)[i];
        auto f = shift(-i) * v.inv();
        return (f.log(m - i *
             k) * k).exp(m - i * k).shift(i * k) * power(v, k);
    }
    Poly sqrt(int m) const {
        Poly x{1};
        int k = 1;
        while (k < m) {
            k *= 2;
            x = (x +
                  (trunc(k) * x.inv(k)).trunc(k)) * CInv<2, P>;
        }
        return x.trunc(m);
    }
    Poly mulT(Poly b) const {
        if (b.size() == 0) return Poly();
        int n = b.size();
        reverse(b.begin(), b.end());
        return ((*this) * b).shift(-(n - 1));
    }
    vector<Value> eval(vector<Value> x) const {
        if (this->size() == 0)
            return vector<Value>(x.size(), 0);
        const int n = max(x.size(), this->size());
        vector<Poly> q(4 * n);
        vector<Value> ans(x.size());
        x.resize(n);
        function<void(
             int, int, int)> build = [&](int p, int l, int r) {
            if (r - l == 1) {
                q[p] = Poly{1, -x[l]};
            } else {
                int m = (l + r) / 2;
                build(2 * p, l, m);
                build(2 * p + 1, m, r);
                q[p] = q[2 * p] * q[2 * p + 1];
            }
        };
        build(1, 0, n);
        function<void(int, int, int, const Poly &)>
             work = [&](int p, int l, int r, const Poly &num) {
            if (r - l == 1) {
                if (l < int(ans.size()))
                    ans[l] = num[0];
            } else {
                int m = (l + r) / 2;
                work(2 * p, l,
                      m, num.mulT(q[2 * p + 1]).resize(m - l));
                work(2 * p + 1,
                      m, r, num.mulT(q[2 * p]).resize(r - m));
            }
        };
        work(1, 0, n, mulT(q[1].inv(n)));
        return ans;
    }
};
template<int P = 998244353>
Poly<P> berlekampMassey(const Poly<P> &s) {
    Poly<P> c, oldC;
    int f = -1;
    for (int i = 0; i < s.size(); i++) {
        auto delta = s[i];
        for (int j = 1; j <= c.size(); j++)
            delta -= c[j - 1] * s[i - j];
        if (delta == 0) continue;
        if (f == -1) {
            c.resize(i + 1);
            f = i;
        } else {
            auto d = oldC;
            d *= -1;
            d.insert(d.begin(), 1);
            Mint<P> df1 = 0;
            for (int j = 1; j <= d.size(); j++)
                df1 += d[j - 1] * s[f + 1 - j];
            assert(df1 != 0);
            auto coef = delta / df1;
            d *= coef;
            Poly<P> zeros(i - f - 1);
            zeros.insert(zeros.end(), d.begin(), d.end());
            d = zeros;
            auto temp = c;
            c += d;
            if (i - temp.size() > f - oldC.size()) {
                oldC = temp;
                f = i;
            }
        }
    }
    c *= -1;
    c.insert(c.begin(), 1);
    return c;
}
template<int P = 998244353>
Mint<P> linearRecurrence(Poly<P> p, Poly<P> q, ll n) {
    int m = q.size() - 1;
    while (n > 0) {
        auto newq = q;
        for (int i = 1; i <= m; i += 2)
            newq[i] *= -1;
        auto newp = p * newq;
        newq = q * newq;
        for (int i = 0; i < m; i++)
            p[i] = newp[i * 2 + n % 2];
        for (int i = 0; i <= m; i++)
            q[i] = newq[i * 2];
        n /= 2;
    }
    return p[0] / q[0];
}
```

12 Else

12.1 Python [7c66a4]

```
from decimal import * # 高精度浮點數
from fractions import * # 分數
from random import *
from math import *
# set decimal prec bigger if it could overflow in precision
setcontext
     (Context(prec=10, Emax=MAX_EMAX, rounding=ROUND_FLOOR))
# read and print
x = int(input())
a, b, c = list(map(Fraction, input().split()))
arr = list(map(Decimal, input().split()))
print(*arr)
# set
st = set(); st.add((a, b)); st.remove((a, b))
if not (a, b) in st:
# dict
d = dict(); d[(a, b)] = 1; del d[(a, b)]
for (a, b) in d.items():
# random
arr = [randint(l, r) for i in range(size)]
choice([8, 6, 4, 1]) # random pick one
shuffle(arr)
```

12.2 Bigint [a11197]

```
struct Bigint { // not support hex division
private:
    using u128 = __uint128_t;
    static const int digit = 9;  // hex: 7
    static const int base = 10;  // hex: 16
    static const int B = power(ll(base), digit);
    Bigint(vector<int> x, int sgn) : x(x), sgn(sgn) {}
    template<class U>
    vector<int> norm(vector<U> a) {
        if (a.empty()) return {0};
        for (int i = 0; i < a.size(); i++) {
            U c = a[i];
            a[i] = c % B;
            c /= B;
            if (c) {
                if (i == a.size() - 1) a.push_back(c);
                else a[i + 1] += c;
            }
        }
        while (a.size() > 1 && a.back() == 0) a.pop_back();
        return {a.begin(), a.end()};
    }
    void resign() {
        sgn = x.back() == 0 ? 1 : sgn;
    }
    vector<int> Add(vector<int> a, vector<int> b) {
        int n = max(a.size(), b.size());
        a.resize(n), b.resize(n);
        for (int i = 0; i < n; i++) a[i] += b[i];
        return norm(a);
    }
    vector<int> Minus(vector<int> a, vector<int> b) {
        int n = max(a.size(), b.size());
        a.resize(n), b.resize(n);
        for (int i = 0; i < n; i++) {
            a[i] -= b[i];
            if (a[i] < 0) a[i] += B, a[i + 1]--;
        }
        return norm(a);
    }
    int toInt(char c) const {
        if (isdigit(c)) return c - '0';
        else return c - 'A' + 10;
    }
    char toChar(int c) const {
        if (c < 10) return c + '0';
        else return c - 10 + 'A';
    }
public:
    int sgn = 1;
    vector<int> x; // 反著存
    Bigint() : x {0}, sgn(1) {}
    Bigint(ll a) {
        *this = Bigint(std::to_string(a));
    }
    Bigint(string s) {
        if (s.empty()) {
            *this = Bigint();
        }
        if (s[0] == '-') s.erase(s.begin()), sgn = -1;
        int add = 0, cnt = 0, b = 1;
        while (s.size()) {
            if (cnt == digit) {
                x.push_back(add), add = cnt = 0;
                b = 1;
            }
            add += toInt(s.back()) * b;
            cnt++, b *= base;
            s.pop_back();
        }
        if (add) x.push_back(add);
        x = norm(x);
    }
    int size() const { return x.size(); }
    Bigint abs() const { return Bigint(x, 1); }
    string to_string() const {
        string res;
        for (int i = 0; i < x.size(); i++) {
            string add;
            int v = x[i];
            for (int j = 0; j < digit; j++)
                add += toChar(v % base), v /= base;
            res += add;
        }
        while (res.size() > 1 && res.back() == '0')
            res.pop_back();
        if (sgn == -1) res += '-';
        reverse(res.begin(), res.end());
        return res;
    }
    Bigint operator-() const { return Bigint(x, -sgn); }
    Bigint &operator+=(const Bigint &rhs) & {
        if (sgn != rhs.sgn) return *this -= (-rhs);
        x = Add(x, rhs.x), resign();
        return *this;
    }
    Bigint &operator-=(const Bigint &rhs) & {
        if (sgn != rhs.sgn) return *this += -rhs;
        if (abs() < rhs.abs()) return *this = -(rhs - *this);
        x = Minus(x, rhs.x), resign();
        return *this;
    }
    friend Bigint operator+(Bigint lhs, Bigint rhs) {
        return lhs += rhs;
    }
    friend Bigint operator-(Bigint lhs, Bigint rhs) {
        return lhs -= rhs;
    }
    friend istream &operator>>(istream &is, Bigint &a) {
        string v; is >> v; a = Bigint(v); return is;
    }
    friend ostream &operator<<(ostream &os, const Bigint &a) {
        os << a.to_string();
        return os;
    }
    friend bool operator<(const Bigint &a, const Bigint &b) {
        if (a.sgn != b.sgn) return a.sgn < b.sgn;
        if (a.x.size() != b.x.size()) {
            return a.x.size() < b.x.size();
        } else {
            for (int i = a.x.size() - 1; i >= 0; i--)
                if (a.x[i] != b.x[i]) return a.x[i] < b.x[i];
        }
        return 0;
    }
    friend bool operator>(const Bigint &a, const Bigint &b) {
        if (a.sgn != b.sgn) return a.sgn > b.sgn;
        if (a.x.size() != b.x.size()) {
            return a.x.size() > b.x.size();
        } else {
            for (int i = a.x.size() - 1; i >= 0; i--)
                if (a.x[i] != b.x[i]) return a.x[i] > b.x[i];
        }
        return 0;
    }
    friend bool operator==(const Bigint &a, const Bigint &b) {
        return a.sgn == b.sgn && a.x == b.x;
    }
    friend bool operator!=(const Bigint &a, const Bigint &b) {
        return a.sgn != b.sgn || a.x != b.x;
    }
    friend bool operator>=(const Bigint &a, const Bigint &b) {
        return a == b || a > b;
    }
    friend bool operator<=(const Bigint &a, const Bigint &b) {
        return a == b || a < b;
    }
};
Bigint abs(const Bigint &a) { return a.abs(); }
Bigint stoBigint(const string &s) { return Bigint(s); }
```

12.3 Multiple [fc8c31]

```
// Require:
// Mint, NTT ~constructor and * operator
const int P1 = 1045430273;
const int P2 = 1051721729;
const int P3 = 1053818881;
const int r12 = Mint<P2>(Mint<P1>::getMod()).inv().x;
const int r13 = Mint<P3>(Mint<P1>::getMod()).inv().x;
const int r23 = Mint<P3>(Mint<P2>::getMod()).inv().x;
const int r1323 = Mint<P3>(ll(r13) * r23).x;
const ll w1 = Mint<P1>::getMod();
const ll w2 = w1 * Mint<P2>::getMod();
// Garner's Algorithm
template <typename T>
vector<T> arbitraryMult
     (const vector<int> &a, const vector<int> &b) {
    int n = a.size(), m = b.size();
    Poly<P1> x = Poly<P1>
         >(a.begin(), a.end()) * Poly<P1>(b.begin(), b.end());
    Poly<P2> y = Poly<P2>
         >(a.begin(), a.end()) * Poly<P2>(b.begin(), b.end());
    Poly<P3> z = Poly<P3>
         >(a.begin(), a.end()) * Poly<P3>(b.begin(), b.end());
    vector<T> res(x.size());
    for (int i = 0; i < x.size(); i++) {
        ll p = x[i].x;
        ll q = (y[i].x + P2 - p) * r12 % P2;
        ll r =
             ((z[i] + P3 - p) * r1323 + (P3 - q) * r23).x % P3;
        res[i] = (T(r) * w2 + q * w1 + p);
    }
    return res;
}
private:
    vector<int> Multiple(vector<int> a, vector<int> b) {
        return norm(arbitraryMult<u128>(a, b));
    }
    vector<int> smallMul(vector<int> a, int v) {
        vector<ll> res(a.begin(), a.end());
        for (auto &x : res) x *= v;
        return norm(res);
    }
public:
    Bigint &operator*=(const Bigint &rhs) & {
        x = rhs.size()
             == 1 ? smallMul(x, rhs.x[0]) : Multiple(x, rhs.x);
        sgn *= rhs.sgn, resign();
        return *this;
    }
    friend Bigint operator*(Bigint lhs, Bigint rhs) {
        return lhs *= rhs;
    }
```

12.4 Division [816dd0]

```
private:
    vector<int> smallDiv(vector<int> a, int v) {
        ll add = 0;
        for (int i = a.size() - 1; i >= 0; i--) {
            add = add * B + a[i];
            int q = add / v;
            a[i] = q, add %= v;
        }
        return norm(a);
    }
    Bigint &operator<<=(int n) & {
        if (!x.empty()) {
            vector<int> add(n, 0);
            x.insert(x.begin(), add.begin(), add.end());
        }
        return *this;
    }
    Bigint &operator>>=(int n) & {
        x = vector
             <int>(x.begin() + min(n, int(x.size())), x.end());
        x = norm(x);
        return *this;
    }
    friend Bigint operator<<(Bigint lhs, int n) {
        return lhs <<= n;
    }
    friend Bigint operator>>(Bigint lhs, int n) {
        return lhs >>= n;
    }
public:
    Bigint &operator/=(const Bigint &rhs) & {
        Bigint a = abs(), b = rhs.abs();
        sgn *= rhs.sgn;
        if (a < b) return *this = Bigint();
        if (b.size() == 1) {
            x = smallDiv(x, rhs.x[0]);
        } else {
            Bigint inv = 1LL * B * B / b.x.back();
            Bigint pre = 0, res = 0;
            int d = a.size() + 1 - b.size();
            int cur = 2, bcur = 1;
            while (inv != pre || bcur < b.size()) {
                bcur = min(bcur << 1, b.size());
                res.x = {b.x.end() - bcur, b.x.end()};
                pre = inv;
                inv *= ((Bigint
                     (2) << (cur + bcur - 1)) - inv * res);
                cur = min(cur << 1, d);
                inv.x = {inv.x.end() - cur, inv.x.end()};
            }
            inv.x = {inv.x.end() - d, inv.x.end()};
            res = a * inv;
            res >>= a.size();
            Bigint mul = res * b;
            while (mul + b <= a) res += 1, mul += b;
            x = norm(res.x);
        }
        return *this;
    }
    Bigint &operator%=(const Bigint &rhs) & {
        return *this = *this - (*this / rhs) * rhs;
    }
    friend Bigint operator/(Bigint lhs, Bigint rhs) {
        return lhs /= rhs;
    }
    friend Bigint operator%(Bigint lhs, Bigint rhs) {
        return lhs %= rhs;
    }
Bigint gcd(Bigint a, Bigint b) {
    while (b != 0) {
        Bigint r = a % b;
        a = b, b = r;
    } return a;
}
```

12.5 Division-Python [110bd8]

```
from decimal import * # 無誤差浮點數
setcontext(
    Context(prec=4000000, Emax=4000000, rounding=ROUND_FLOOR))
t = int(input())
for i in range(t):
    a, b = map(Decimal, input().split())
    d, m = divmod(a, b)
    print(d, m)
```