Contents

1 Basic 1
1.1 Compare Fuction 1
1.2 Pbds 1
1.3 Double 1
1.4 Int128 1
1.5 Rng 1

2 Graph 1
2.1 DFS And BFS 1
2.2 Prim 2
2.3 Bellman-Ford 2
2.4 Floyd-Warshall 2
2.5 Euler 2
2.6 DSU 2
2.7 SCC 3
2.8 VBCC 3
2.9 EBCC 3
2.10 2-SAT 4
2.11 Functional Graph 4

3 Data Structure 4
3.1 Segment Tree 4
3.2 Fenwick 5
3.3 Range Fenwick 5
3.4 Persistent Segment Tree . 6
3.5 Treap 6
3.6 RMQ 7
3.7 Mo 7

4 Flow Matching 7
4.1 Dinic 7
4.2 Min Cut 7
4.3 MCMF 8
4.4 Hungarian 8
4.5 Theorem 8

5 String 8
5.1 Hash 8
5.2 KMP 9
5.3 Z Function 9
5.4 Manacher 9
5.5 Trie 9
5.6 SA 9
5.7 SAM 9
5.8 Palindrome Tree 10
5.9 Duval 10

6 Math 11
6.1 Mint 11
6.2 Combination 11
6.3 Sieve 11
6.4 Miller Rabin Pollard Rho . 11
6.5 CRT 12
6.6 Matrix 12
6.7 Mex 12
6.8 Game Theorem 12
6.9 Fraction 12
6.10 Gaussian Elimination . . . 13
6.11 Integer Partition 13
6.12 Mobius Theorem 13
6.13 Mobius Inverse 13
6.14 Catalan Theorem 14
6.15 Burnside's Lemma 14

7 Search and Gready 14
7.1 Binary Search 14

8 Tree 14
8.1 Binary Lifting LCA 14
8.2 Centroid Decomposition . 14
8.3 Heavy Light Decomposition 14
8.4 Link Cut Tree 15
8.5 Virtual Tree 16
8.6 Dominator Tree 16

9 DP 16
9.1 LCS 16
9.2 LIS 17
9.3 Edit Distance 17
9.4 Bitmask 17
9.5 Projects 17
9.6 Removal Game 17
9.7 Monotonic Queue 18
9.8 SOS 18
9.9 CHT 18
9.10 DNC 18
9.11 LiChao Segment Tree . . . 19
9.12 Codeforces Example . . . 19

10 Geometry 19
10.1 Basic 19
10.2 Min Euclidean Distance . . 21
10.3 Max Euclidean Distance . . 21
10.4 Lattice Points 21
10.5 Min Circle Cover 21
10.6 Min Rectangle Cover . . . 22

11 Polynomial 22
11.1 FFT 22
11.2 NTT 22

12 Else 24
12.1 Python 24
12.2 Bigint 24
12.3 Multiple 25
12.4 Division 25
12.5 Division-Python 25

1 Basic

1.1 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);
```

1.2 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>;
```

1.3 Double [93fa38]

```
struct D {
    double x;
    D(double x = 0.0) : x{x} {}
    constexpr static double eps = 1E-12;
    explicit operator double() const { return x; }
    D operator-() const {
        return D(-x);
    }
    D &operator+=(D rhs) & {
        x += rhs.x; return *this;
    }
    D &operator-=(D rhs) & {
        x -= rhs.x; return *this;
    }
    D &operator*=(D rhs) & {
        x *= rhs.x; return *this;
    }
    D &operator/=(D rhs) & {
        assert(fabs(rhs.x) > eps);
        x /= rhs.x; return *this;
    }
    friend D operator+(D lhs, D rhs) {
        return lhs += rhs;
    }
    friend D operator-(D lhs, D rhs) {
        return lhs -= rhs;
    }
    friend D operator*(D lhs, D rhs) {
        return lhs *= rhs;
    }
    friend D operator/(D lhs, D rhs) {
        return lhs /= rhs;
    }
    friend istream &operator>>(istream &is, D &a) {
        double v; is >> v; a = D(v); return is;
    }
    friend ostream &operator<<(ostream &os, const D &a) {
        return os << fixed << setprecision(10)
                  << a.x + (a.x > 0 ? eps : a.x < 0 ? -eps : 0);
    } // eps should < precision
    friend bool operator<(D lhs, D rhs) {
        return lhs.x - rhs.x < -eps;
    }
    friend bool operator>(D lhs, D rhs) {
        return lhs.x - rhs.x > eps;
    }
    friend bool operator==(D lhs, D rhs) {
        return fabs(lhs.x - rhs.x) < eps;
    }
};
```

1.4 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;
}
```

1.5 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 DFS And BFS [1f02d8]

```
void dfsBfs() {
    int n;
    vector<vector<int>> adj(n);
    // dfs_graph
    vector<bool> vis(n);
    auto dfs = [&](auto self, int u) -> void {
        if (vis[u]) return;
        vis[u] = true;
        for (auto v: adj[u]) {
            self(self, v);
        }
    };
    dfs(dfs, 0);
    // bfs
```

```
    vector<int> dep(n, -1);
    auto bfs = [&](auto self, int s) -> void {
        queue<int> que;
        dep[s] = 0, que.push(s);
        while (!que.empty()) {
            int u = que.front(); que.pop();
            for (auto v : adj[u]) {
                if (dep[v] == -1) {
                    dep[v] = dep[u] + 1;
                    que.push(v);
                }
            }
        }
    };
    bfs(bfs, 0);
}
```

2.2 Prim [7e2d87]

```
auto prim =
     [&](int n, vector<vector<pair<int, int>>> &adj) -> bool {
    int sz = 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 [u, w] = pq.top();
        pq.pop();
        if (vis[u]) continue;
        vis[u] = true;
        sz++;
        for (auto v : adj[u])
            if (!vis[v.first])
                pq.emplace(v.second, v.first);
    }
    if (sz == n) return true;
    return false;
};
```

2.3 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 << " ";
}
```

2.4 Floyd-Warshall [2f66b9]

```
constexpr 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];
}
```

2.5 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());
```

2.6 DSU [b7ac4a]

```
struct DSU {
    int n;
    vector<int> boss, siz;
    DSU(int n_ = 0) { init(n_); }
    void init(int n_) {
        n = n_; boss.resize(n);
        iota(boss.begin(), boss.end(), 0);
        siz.assign(n, 1);
    }
    int find(int x) {
        if (boss[x] == x) return x;
        return boss[x] = find(boss[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];
        boss[y] = x;
        n--;
        return true;
    }
    int size(int x) {
        return siz[find(x)];
    }
};
struct DSU {
    int n;
    vector<int> boss, siz, stk;
    DSU(int n_ = 0) { init(n_); }
    void init(int n_) {
        n = n_;
        boss.resize(n);
        iota(boss.begin(), boss.end(), 0);
        siz.assign(n, 1);
        stk.clear();
    }
    int find(int x) {
        return x == boss[x] ? x : find(boss[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];
        boss[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[boss[y]] -= siz[y];
            boss[y] = y;
        }
    }
    int size(int x) {
```

```
        return siz[find(x)];
    }
};
```

2.7 SCC [26d711]

```
struct SCC {
    int n, cur, cnt;
    vector<vector<int>> adj;
    vector<int> stk, dfn, low, bel;
    SCC(int n_ = 0) { init(n_); }
    void init(int n_) {
        n = n_;
        adj.assign(n, {});
        dfn.assign(n, -1), low.resize(n);
        bel.assign(n, -1), stk.clear();
        cur = cnt = 0;
    }
    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;
    }
};
```

2.8 VBCC [2d1f9d]

```
struct VBCC {
    int n, cur, cnt;
    vector<vector<int>> adj, bcc;
    vector<int> stk, dfn, low;
    vector<bool> ap;
    VBCC(int n_ = 0) { init(n_); }
    void init(int n_) {
        n = n_;
        adj.assign(n, {});
        dfn.assign(n, -1), low.resize(n);
        bcc.assign(n, {}), ap.assign(n, false);
        stk.clear();
        cur = cnt = 0;
    }
    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 child = 0;
        for (auto y : adj[x]) {
            if (y == p) continue;
            if (dfn[y] == -1) {
                dfs(y, x), child++;
                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 && child > 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(cnt);
        g.cnte.resize(cnt);
        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;
    }
};
```

2.9 EBCC [9d70fc]

```
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_ = 0) { init(n_); }
    void init(int n_) {
        n = n_;
        adj.assign(n, {});
        dfn.assign(n, -1), low.resize(n);
        bel.assign(n, -1), stk.clear();
        bridges.clear();
        cur = cnt = 0;
    }
    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;
    }
};
```

2.10 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; }
};
```

2.11 Functional Graph [e8fd64]

```
constexpr int N = 2E5 + 5;
int cht[N][31]; // 倍增表, 放外面不然 TLE
struct FuntionalGraph {
    int n, cnt;
    vector<int> g, bel, id, len, in, top;
    FuntionalGraph() : n(0) {}
    FuntionalGraph(vector<int> g_) { init(g_); }
    void init(vector<int> g_) {
        n = g_.size(); cnt = 0;
        g = g_; bel.assign(n, -1);
        id.resize(n); len.clear();
        in.assign(n, 0); top.assign(n, -1);
        build();
    }
    void build() {
        for (int i = 0; i < n; i++) {
            cht[i][0] = g[i];
            in[g[i]]++;
        }
        for (int i = 1; i <= 30; i++)
            for (int u = 0; u < n; u++)
                cht[u][i] = cht[cht[u][i - 1]][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;
        }
        if (!cyc.empty())
            ++cnt, len.push_back(cyc.size());
        for (int i = p.size() - 1; i > 0; i--)
            id[p[i - 1]] = id[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;
    */
    SegmentTree() : n(0) {}
    SegmentTree(int n_, Info v_ = Info()) {
        init(n_, v_);
    }

    template<class T>
    SegmentTree(vector<T> init_) {
        init(init_);
    }

    void init(int n_, Info v_ = Info()) {
        init(vector(n_, v_));
    }

    template<class T>
    void init(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(p * 2, l, m);
            build(p * 2 + 1, m, r);
            pull(p);
        };
        build(1, 0, n);
    }
    void pull(int p) {
        info[p] = info[p * 2] + info[p * 2 + 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(p * 2, l, m, tag[p]);
            apply(p * 2 + 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(p *
             2, l, m, ql, qr) + query(p * 2 + 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(p * 2, l, m, ql, qr, v);
        rangeApply(p * 2 + 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 {
    bool set_val = false;
    int add = 0;
    void apply(const Tag& t) & {
        if (t.set_val) {
            set_val = t.set_val;
            add = t.add;
        }
        else {
            add += t.add;
        }
    }
};
*/
struct Info {
    ll sum = 0;
    /*
    void apply(int l, int r, const Tag &t) & {
        if (t.set_val) {
            sum = (r - l) * t.set_val;
        }
        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.n = a.n + b.n;
    c.sum = a.sum + b.sum;
    return c;
}
```

3.2 Fenwick [d41d8c]

```
template<class T>
struct Fenwick { // 全部以 0 based 使用
    int n; vector<T> a;
    Fenwick(int n_ = 0) {
        init(n_);
    }
    void init(int n_) {
        n = n_;
        a.assign(n, T{});
    }
    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) { // 左閉右開查詢
        T ans{};
        for (int i = x; i > 0; i -= i & -i)
            ans = ans + a[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 && cur + a[x + i - 1] <= k) {
                x += i;
                cur = cur + a[x - 1];
            }
        }
        return x;
    }
};
template<class T>
struct TwoDFenwick {  // 全部以 0 based 使用
    int nx, ny;  // row, col 個數
    vector<vector<T>> a;
    TwoDFenwick(int nx_ = 0, int ny_ = 0) {
        init(nx_, ny_);
    }
    void init(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) { // 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 + a[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);
    }
};
```

3.3 Range Fenwick [d41d8c]

```
template<class T>
struct RangeFenwick { // 全部以 0 based 使用
    int n;
    vector<T> d, di;
    RangeFenwick(int n_ = 0) {
        init(n_);
    }
    void init(int n_) {
        n = n_;
        d.assign(n, T{});
        di.assign(n, T{});
    }
    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 nx_ = 0, int ny_ = 0) {
        init(nx_, ny_);
    }
    void init(int nx_, int ny_) {
        nx = nx_; ny = ny_;
        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);
    }
};
```

3.4 Persistent Segment Tree [d41d8c]

```
template<class Info>
struct PST {
    struct Node {
        Info info = Info();
        int lc = 0, rc = 0;
    };
    int n = 0;
    vector<Node> nd;
    vector<int> rt;
    PST() : n(0) {}
    PST(int n_, Info v_ = Info()) {
        init(n_, v_);
    }

    template<class T>
    PST(vector<T> init_) {
        init(init_);
    }

    void init(int n_, Info v_ = Info()) {
        init(vector<Info>(n_, v_));
    }

    template<class T>
    void init(vector<T> init_) {
        n = init_.size();
        nd.clear(); rt.clear();
        nd.emplace_back(); // 讓 root 指向 1-based
        rt.push_back(build(0, n, init_));
    }
    int build(int l, int r, vector<Info> &init_) {
        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, init_);
        nd[id].rc = build(m, r, init_);
        pull(nd[id]);
        return id;
    }
    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) >> 1;
        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 pos, const Info &val) {
        if (int(rt.size()) <= ver) rt.resize(ver + 1);
        rt[ver] = modify(rt[ver], 0, n, pos, val);
    }
    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) >> 1;
        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 {
    int sum = 0;
};
Info operator+(const Info &a, const Info &b) {
    return { a.sum + b.sum };
}
```

3.5 Treap [d41d8c]

```
struct Treap {
    Treap *lc, *rc;
    int pri, siz; bool rev_valid;
    int val; int min;
    Treap(int val_) {
        min = val = val_;
        pri = rand();
        lc = rc = nullptr;
        siz = 1; rev_valid = 0;
    }
    void pull() { // update siz or other information
        siz = 1;
        min = val;
        for (auto c : {lc, rc}) {
            if (!c) continue;
            siz += c->siz;
            min = std::min(min, c->min);
        }
    }
    void push() {
        if (rev_valid) {
            swap(lc, rc);
            if (lc) lc->rev_valid ^= 1;
            if (rc) rc->rev_valid ^= 1;
```

```
        }
        rev_valid = false;
    }
    int find(int k) { // 找到 min 是 k 的位置 (1-based)
        push();
        int ls = (lc ? lc->siz : 0) + 1;
        if (val == k) return ls;
        if (lc && lc->min == k) return lc->find(k);
        else return rc->find(k) + ls;
    }
};
int size(Treap *t) {
    return t ? t->siz : 0;
}
Treap *merge(Treap *a, Treap *b) {
    if (!a || !b) return a ? a : b;
    a->push(); b->push();
    if (a->pri > b->pri) {
        a->rc = merge(a->rc, b);
        a->pull();
        return a;
    }
    else {
        b->lc = merge(a, b->lc);
        b->pull();
        return b;
    }
}
pair<Treap*, Treap*> split(Treap *t, int k) {
    // 分割前 k 個在 first，剩下的在 second
    if (t == nullptr) return {nullptr, nullptr};
    t->push();
    if (size(t->lc) < k) {
        auto [a, b] = split(t->rc, k - size(t->lc) - 1);
        t->rc = a;
        t->pull();
        return {t, b};
    }
    else {
        auto [a, b] = split(t->lc, k);
        t->lc = b;
        t->pull();
        return {a, t};
    }
}
void Print(Treap *t) {
    if (!t) return;
    t->push();
    Print(t->lc);
    cout << t->val;
    Print(t->rc);
}
```

3.6 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);
    }
};
```

3.7 Mo [d41d8c]

```
struct Query {
    int l, r, id;
};
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;
    });
}
```

4 Flow Matching

4.1 Dinic [d41d8c]

```
template<class T>
struct Dinic {
    struct _Edge {
        int to;
        T f, cap; // 流量跟容量
    };
    int n, m, s, t;
    const T INF_FLOW = 1LL << 60;
    vector<vector<int>> g;
    vector<_Edge> e;
    vector<int> h, cur;
    Dinic(int n_ = 0) { init(n_); }
    void init(int n_) {
        n = n_; m = 0;
        h.resize(n); cur.resize(n);
        g.assign(n, {});
        e.clear();
    }
    void add_edge(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 reset() {
        for (int i = 0; i < m; i++) e[i].f = 0;
    }
    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, m; cin >> n >> m;
    Dinic<int> g(n);
    for (int i = 0; i < m; i++) {
        int u, v, cap = 1;
        cin >> u >> v;
        u--; v--;
        g.add_edge(u, v, cap);
        g.add_edge(v, u, cap);
    }
    int res = g.work(0, n - 1);
    cout << res << "\n";
    if (res == 0) return;

    vector<int> vis(n);
    auto find = [&](auto self, int u) -> void {
```

```
        if (!vis[u]) {
            vis[u] = 1;
            for (int id : g.adj[u]) {
                auto e = g.edges[id];
                if (e.cap - e.flow > 0) {
                    self(self, e.to);
                }
            }
        }
    };
    find(find, 0);
    for (int i = 0; i < n; i++) {
        if (!vis[i]) continue;
        for (int id : g.adj[i]) {
            if (id & 1) continue;
            auto e = g.edges[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 = 1 << 30;
    const Tc INF_COST = 1 << 30;
    vector<_Edge> e;
    vector<vector<int>> g;
    vector<Tc> dis;
    vector<int> rt, inq;
    MCMF(int n_ = 0) { init(n_); }
    void init(int n_) {
        n = n_; m = 0;
        e.clear();
        g.assign(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() {
        dis.assign(n, INF_COST);
        rt.assign(n, -1), inq.assign(n, 0);
        queue<int> q; q.push(s);
        dis[s] = 0;
        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;
                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;
    }
    // 限定 flow, 最小化 cost
    pair<Tf, Tc> workFlow(int s_, int t_, Tf need) {
        s = s_, t = t_;
        Tf flow{}; Tc cost{};
        while (spfa()) {
            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];
            if (need == 0) break;
        }
        return {flow, cost};
    }
    // 限定 cost, 最大化 flow
    pair<Tf, Tc> workBudget(int s_, int t_, Tc budget) {
        s = s_, t = t_;
        Tf flow{}; Tc cost{};
        while (spfa()) {
            Tf f = budget / dis[t];
            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, budget -= f * dis[t];
            cost += f * dis[t];
            if (budget == 0 || f == 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_ = 0, int m_ = 0) {
        init(n_, m_);
    }
    void init(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 [7a28d1]

```
constexpr int B = 59;
vector<Z> hash(string &s) {
    vector<Z> ans {0};
    for (auto c : s) {
        ans.push_back(ans.back() * B + (c - 'a' + 1));
    }
    return ans;
}
void solve() {
    string s, sub;
    cin >> s >> sub;
    auto a = hash(s);
    auto q = hash(sub);
    auto find = q.back();
```

```
    int ans = 0;
    int l = 1, r = sub.size(), len = sub.size();
    while (r <= s.size()) {
        if (a[r] - a[l - 1] * power(Z(B), len) == find) {
            ans++;
        }
        l++, r++;
    }
    cout << ans << "\n";
}
```

5.2 KMP [731acf]

```
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() {}
    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 [958661]

```
// 找到對於每個位置的廻文半徑
vector<int> manacher(const string &s) {
    string t = "#";
    for (auto c : s) {
        t += c;
        t += '#';
    }
    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 [72392f]

```
constexpr int N = 1E7;
int tot = 0;
int trie[N][26], cnt[N];
void reset() {
    tot = 0, fill_n(trie[0], 26, 0);
}
int newNode() {
    int x = ++tot;
    cnt[x] = 0, fill_n(trie[x], 26, 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 [c9e6e0]

```
struct SAM {
    // 1 -> 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
    // range-> [len(link) + 1, len]
    struct Node {
        int len, link, fpos;
        array<int, ALPHABET_SIZE> next;
        Node() : len{}, link{}, fpos{}, next{} {}
    };
    vector<Node> t;
    SAM() { init(); }
    void init() {
        t.assign(2, Node());
        t[0].len = -1;
    }
    int newNode() {
        t.emplace_back();
        return t.size() - 1;
    }
    int extend(int p, int c) {
        if (!p) t[p].next[c] = 1;
        if (t[p].next[c]) {
            int q = t[p].next[c];
            if (t[q].len == t[p].len + 1) {
                return q;
            }
            int r = newNode();
            t[r] = t[q];
            t[r].len = t[p].len + 1;
            t[q].link = r;
            while (t[p].next[c] == q) {
                t[p].next[c] = r;
                p = t[p].link;
            }
            return r;
        }
        int cur = newNode();
        t[cur].len = t[p].len + 1;
        t[cur].fpos = t[p].len;
        while (!t[p].next[c]) {
            t[p].next[c] = cur;
            p = t[p].link;
        }
        t[cur].link = extend(p, c);
        // distinct substr += t[cur].len - t[t[cur].link].len;
        return cur;
    }
};
void solve() {  // Substring Order II: build
    string s; cin >> s;
    int n = s.length();
    vector<int> last(n + 1); // s[i - 1] 的後綴終點位置
    last[0] = 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();
    // without this part for distinct substr
    vector<int> cnt(sz);
    // endpos size: substr occurence
    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].len].push_back(i);
    for (int i = n; i > 0; i--)
        for (int u : g[i])
            cnt[sam.t[u].link] += cnt[u];


    vector<ll> dp(sz, -1);
    auto rec = [&](auto self, int u) -> ll {
        if (dp[u] != -1) return dp[u];
        dp[u] = cnt[u]; // = 1 for distinct
        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, 1);

    int k, p = 1; cin >> k;
    string ans;
    while (k > 0) {
        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--, p = v;
                    break;
                }
            }
        }
    }
    cout << ans << "\n";
}
```

5.8 Palindrome Tree [52fd3d]

```
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() { init(); }
    void init() {
        s.clear();
        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 [86ac44]

```
// 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();
    int 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 [6eb719]

```
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; b /= 2, a *= a)
        if (b & 1) res *= a;
    return res;
}
template<int P>
struct Mint {
    // Dynamic Mint, not necessary
    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) & {
        x = norm(x + rhs.x);
        return *this;
    }
    Mint &operator-=(Mint rhs) & {
        x = norm(x - rhs.x);
        return *this;
    }
    Mint &operator*=(Mint rhs) & {
        x = x * rhs.x % getMod();
        return *this;
    }
    Mint &operator/=(Mint rhs) & {
        return *this *= rhs.inv();
    }
    friend Mint operator+(Mint lhs, Mint rhs) {
        return lhs += rhs;
    }
    friend Mint operator-(Mint lhs, Mint rhs) {
        return lhs -= rhs;
    }
    friend Mint operator*(Mint lhs, Mint rhs) {
        return lhs *= rhs;
    }
    friend Mint operator/(Mint lhs, Mint rhs) {
        return lhs /= rhs;
    }
    friend istream &operator>>(istream &is, Mint &a) {
        ll v; is >> v; a = Mint(v); return is;
    }
    friend ostream &operator<<(ostream &os, const Mint &a) {
        return os << a.x;
    }
    // following operators are not necessary
    friend bool operator==(Mint lhs, Mint rhs) {
        return lhs.x == rhs.x;
    }
    friend bool operator!=(Mint lhs, Mint rhs) {
        return lhs.x != rhs.x;
    }
    friend bool operator<(Mint lhs, Mint rhs) {
        return lhs.x < rhs.x;
    }
};
template<>
int Mint<0>::Mod = 998244353;
constexpr int P = 1E9 + 7;
using Z = Mint<P>;
```

6.2 Combination [f12983]

```
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 [6b1b59]

```
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (!b) {
        x = 1, y = 0;
        return a;
    }
    ll g = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return g;
}
ll inv(ll x, ll m) {
    ll a, b;
    exgcd(x, m, a, b);
    a %= m;
    if (a < 0) a += m;
    return a;
}
// gcd(mod) = 1, res % mod_i = remain_i
// a: remain, mod
ll CRT(vector<pair<ll, ll>> &a) {
    ll s = 1, res = 0;
    for (auto [r, m] : a) s *= m;
    for (auto [r, m] : a) {
        ll t = s / m;
        res += r * t % s * inv(t, m) % s;
        if (res >= s) res -= s;
    }
    return res;
}
```

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 [14628f]

```
template<class T>
int 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: $f(0) = 0, f(1) = mex(f(0)) = 1, f(2) = mex(f(0), f(1)) = 2 \ldots$, 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 $sg(1)g(2)\ldots g(n)$, 就可以用 xor 性質處理不連續組合

6.9 Fraction [3f8970]

```
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) {
        istringstream ss(str);
        char slash;
        if (str.find('/') != -1) {
            ss >> n >> slash >> d;
        } else {
            ss >> n;
            d = 1;
        }
        Fraction(n, d);
    }
    Fraction operator+=(Fraction rhs) & {
        n = n * rhs.d + rhs.n * d;
        d *= rhs.d;
        reduce();
        return *this;
    }
    Fraction operator-=(Fraction rhs) & {
        n = n * rhs.d - rhs.n * d;
        d *= rhs.d;
        reduce();
        return *this;
    }
    Fraction operator*=(Fraction rhs) & {
        n *= rhs.n;
        d *= rhs.d;
        reduce();
        return *this;
    }
    Fraction operator/=(Fraction rhs) & {
        assert(rhs.n != 0);
        n *= rhs.d;
        d *= rhs.n;
        reduce();
        return *this;
    }
    friend Fraction operator+(Fraction lhs, Fraction rhs) {
        return lhs += rhs;
    }
    friend Fraction operator-(Fraction lhs, Fraction rhs) {
        return lhs -= rhs;
    }
    friend Fraction operator*(Fraction lhs, Fraction rhs) {
        return lhs *= rhs;
    }
    friend Fraction operator/(Fraction lhs, Fraction rhs) {
        return lhs /= rhs;
    }
    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;
    }
    friend bool operator==(Fraction lhs, Fraction rhs) {
        return lhs.n * rhs.d == rhs.n * lhs.d;
    }
    friend bool operator!=(Fraction lhs, Fraction rhs) {
        return lhs.n * rhs.d != rhs.n * lhs.d;
    }
    friend bool operator<(Fraction lhs, Fraction rhs) {
        return lhs.n * rhs.d < rhs.n * lhs.d;
    }
};
```

6.10 Gaussian Elimination [a5e69e]

```
// 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 [a2c848]

```
// 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;
    ll 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 $\sum_{i=1}^{n} f(i)g(\lfloor \frac{n}{i} \rfloor)$ 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

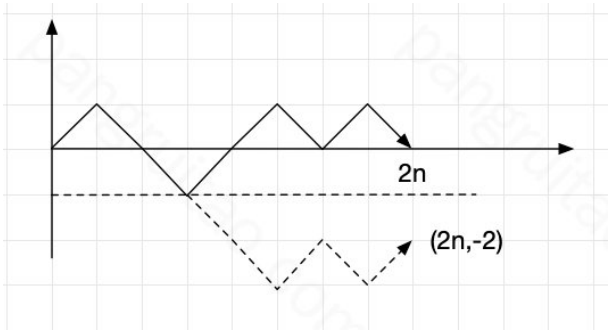
$$\begin{aligned}&\sum_{i=a}^{b}\sum_{j=c}^{d}[gcd(i,j)=k]\\ &\Rightarrow\sum_{i=1}^{x}\sum_{j=1}^{y}[gcd(i,j)=k]\\ &=\sum_{i=1}^{\lfloor\frac{x}{k}\rfloor}\sum_{j=1}^{\lfloor\frac{y}{k}\rfloor}\epsilon(gcd(i,j))\\ &=\sum_{i=1}^{\lfloor\frac{x}{k}\rfloor}\sum_{j=1}^{\lfloor\frac{y}{k}\rfloor}\sum_{d|gcd(i,j)}\mu(d)\\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

6.13 Mobius Inverse [d41d8c]

```
const int maxn = 2E5;
ll mobiusPref[maxn];
void init() {
    mobiusPref[1] = 1;
    vector<ll> wei
        (maxn);  // wei = 0 代表是質數，-1 代表可被平方數整除
    for (ll i = 2; i < maxn; i++) {
        if (wei[i] == -1) {
            mobiusPref[i] = mobiusPref[i - 1];
            continue; // 包含平方
        }
        if (wei[i] == 0) {
            wei[i] = 1;
            for (ll j = 2; i * j < maxn; j++) {
                if (j % i == 0) wei[i * j] = -1;
                else if (wei[i * j] != -1) wei[i * j]++;
            }
        }
        mobiusPref[i]
            = mobiusPref[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 += (mobiusPref[r] - mobiusPref[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 $\frac{(2n)!}{n!n!}=C_n^{2n}$
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_{n-1}^{2n} 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
    while (lo < hi) {
        int x = (lo + hi + 1) / 2;
        if (check(x)) lo = x;
        else hi = x - 1;
    }
    cout << lo; // 保證有解

    while (lo <= hi) {
        int x = (lo + hi) / 2;
        if (check(x)) lo = x + 1;
        else hi = x - 1;
    }
    cout << hi; // 範圍外代表無解

    // 二分找下界
    while (lo < hi) {
        int x = (lo + hi) / 2;
        if (check(m)) hi = x;
        else lo = x + 1;
    }
    cout << lo; // 保證有解

    while (lo <= hi) {
        int x = (lo + hi) / 2;
        if (check(m)) hi = x - 1;
        else lo = x + 1;
    }
    cout << lo; // 範圍外代表無解
}
```

8 Tree

8.1 Binary Lifting LCA [5af658]

```
const int Lg = 20; // __lg(max(n, qi)), [0, Lg]
vector<vector<int>> up;
vector<int> dep, dfn;
void build(int n, vector<vector<int>> &g, int rt = 0) {
    up.assign(n, vector<int>(Lg + 1));
    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++)
            up[x][i] = up[up[x][i - 1]][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;
}
```

8.2 Centroid Decomposition [4f8836]

```
struct CentriodDecomposition {
    int n;
    vector<vector<int>> adj;
    vector<bool> vis;
    vector<int> siz;
    CentriodDecomposition(int n_ = 0) { init(n_); }
    void init(int n_) {
        n = n_;
        adj.assign(n, {});
        vis.assign(n, false);
        siz.assign(n, 1);
    }
    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    void getSiz(int x, int p = -1) {
        siz[x] = 1;
        for (int y : adj[x]) {
            if (y == p || vis[y]) continue;
            getSiz(y, x);
            siz[x] += siz[y];
        }
    }
    int getCen(int x, int sz, int p = -1) {
        for (int y : adj[x]) {
            if (y == p || vis[y]) continue;
            if (siz[y] * 2 > sz)
                return getCen(y, sz, x);
        }
        return x;
    }
    void getAns(int x, int p) {
        // do something
        for (int y : adj[x]) {
            if (y == p || vis[y]) continue;
            getAns(y, x);
        }
    }
    void work(int x = 0) {
        getSiz(0, x);
        int cen = getCen(x, siz[x]);
        vis[cen] = true;
        for (int y : adj[cen]) {
            if (vis[y]) continue;
            getAns(y, cen);
        }
        for (int y : adj[cen]) {
            if (vis[y]) continue;
            work(y);
        }
    }
};
```

8.3 Heavy Light Decomposition [41d99e]

```
struct HLD {
    int n, cur;
    vector<int> siz, top, dep, parent, in, out, seq;
    vector<vector<int>> adj;
    HLD(int n_ = 0) { init(n_); }
    void init(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) {
        swap(rt, v);
        if (rt == v) return rt;
        if (!isAncester(rt, v)) return parent[rt];
        auto it = upper_bound(adj[
            rt].begin(), adj[rt].end(), v, [&](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 [510da5]

```
template<class Info, class Tag>
struct LinkCutTree { // 1-based
    struct Node {
        Info info = Info();
        Tag tag = Tag();
        bool rev = false;
        int size = 0;
        int ch[2], p = 0;
    };
    vector<Node> nd;
    LinkCutTree(int n = 0) { init(n); }
    void init(int n) {
        nd.clear();
        nd.emplace_back();
        resize(n);
    }
    void resize(int n) {
        nd.resize(n + 1);
    }
    bool isrt(int t) {
        return !nd[t].p || (
            nd[nd[t].p].ch[0] != t && nd[nd[t].p].ch[1] != t);
    }
    void makeRev(int t) {
        swap(nd[t].ch[0], nd[t].ch[1]);
        nd[t].rev ^= true;
    }
    void apply(int t, const Tag &v) {
        nd[t].info.apply(nd[t].size, v);
        nd[t].tag.apply(v);
    }
    void push(int t) {
        if (nd[t].rev) {
            if (nd[t].ch[0]) makeRev(nd[t].ch[0]);
            if (nd[t].ch[1]) makeRev(nd[t].ch[1]);
            nd[t].rev = false;
        }
        if (nd[t].ch[0]) apply(nd[t].ch[0], nd[t].tag);
        if (nd[t].ch[1]) apply(nd[t].ch[1], nd[t].tag);
        nd[t].tag = Tag();
    }
    void pull(int t) {
        nd[t].size
            = 1 + nd[nd[t].ch[0]].size + nd[nd[t].ch[1]].size;
        nd[t].info
            .pull(nd[nd[t].ch[0]].info, nd[nd[t].ch[1]].info);
    }
    int pos(int t) {
        return nd[nd[t].p].ch[1] == t;
    }
    void pushAll(int t) {
        if (!isrt(t)) pushAll(nd[t].p);
        push(t);
    }
    void rotate(int t) {
        int q = nd[t].p, x = !pos(t);
        nd[q].ch[!x] = nd[t].ch[x];
        if (nd[t].ch[x]) nd[nd[t].ch[x]].p = q;
        nd[t].p = nd[q].p;
        if (!isrt(q)) nd[nd[q].p].ch[pos(q)] = t;
        nd[t].ch[x] = q, nd[q].p = t;
        pull(q);
    }
    void splay(int t) {
        pushAll(t);
        while (!isrt(t)) {
            if (!isrt(nd[t].p)) {
                if (pos(t) == pos(nd[t].p)) {
                    rotate(nd[t].p);
                } else {
                    rotate(t);
                }
            }
            rotate(t);
        }
        pull(t);
    }
    void access(int t) { // access 後自動 splay
        for (int i = t, q = 0; i; q = i, i = nd[i].p) {
            splay(i);
            nd[i].ch[1] = q;
            pull(i);
        }
        splay(t);
    }
    void makeRoot(int t) {
        access(t), makeRev(t);
    }
    int findRoot(int t) {
        access(t);
        int x = t;
        while (nd[x].ch[0]) {
            push(x);
            x = nd[x].ch[0];
        }
        access(x);
        return x;
    }
    bool connected(int x, int y) {
        return findRoot(x) == findRoot(y);
    }
    bool neighber(int x, int y) {
        makeRoot(x), access(y);
        if (nd[y].ch[0] != x || nd[x].ch[1]) return false;
        return true;
    }
    void split(int rt, int y) {
        makeRoot(y), access(rt);
    }
    void link(int x, int y) {
        makeRoot(x);
        if (findRoot(y) != x) nd[x].p = y;
    }
    void cut(int x, int y) {
        makeRoot(x), access(y);
        nd[y].ch[0] = nd[nd[y].ch[0]].p = 0;
        pull(x), pull(y);
    }
    void modify(int x, const Info &v) {
        access(x);
        nd[x].info = v;
    }
```

```
    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 nd[x].info;
    }
};
constexpr int Mod = 51061;
struct Tag {
    ll add = 0; ll mul = 1;
    void apply(const Tag &v) {
        mul = mul * v.mul % Mod;
        add = (add * v.mul % Mod + v.add) % Mod;
    }
};
struct Info {
    ll val = 0; ll sum = 0;
    void apply(int size, const Tag &v) {
        val = (val * v.mul % Mod + v.add) % Mod;
        sum = (sum * v.mul % Mod + v.add * size % Mod) % Mod;
    }
    void pull(const Info &l, const Info &r) {
        sum = (l.sum + r.sum + val) % Mod;
    }
};
```

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 [1babd0]

```
// 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_ = 0) { init(n_); }
    void init(int n_) {
        n = n_, id = 0;
        adj.assign(n, {});
        radj.assign(n, {});
        bucket.assign(n, {});
        sdom.resize(n), dom.assign(n, -1);
        vis.assign(n, -1), rev.resize(n);
        pa.resize(n), rt.resize(n);
        mn.resize(n), res.resize(n);
    }
    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 [6ef49c]

```
void LCS() {
    int m, n; cin >> m >> n;
    string s1, s2; cin >> s1 >> s2;
    int L = 0;
    vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));
    for (int i = 1; i <= m; i++) {
```

```
        for (int j = 1; j <= n; j++) {
            if (s1[i - 1] == s2[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 length = dp[m][n];
    cout << length << "\n";
    string s(length, 'c'); // backtracking
    while (m >= 1 && n >= 1) {
        if (s1[m - 1] == s2[n - 1]) {
            s[length - 1] = s1[m - 1];
            m--, n--, length--;
        }
        else {
            if (dp[m - 1][n] > dp[m][n - 1]) m--;
            else n--;
        }
    }
    cout << s << "\n";
}
```

9.2 LIS [2b086e]

```
void LIS() {
    int n; cin >> n;
    vector<int> v(n);
    for (int i = 0; i < n; i++) cin >> v[i];
    int dp[n], L = 1;
    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 {
            auto it
                 = lower_bound(stk.begin(), stk.end(), v[i]);
            *it = v[i]; dp[i] = it - stk.begin() + 1;
        }
    }
    vector<int> ans; cout << L << "\n";
    for (int i = n - 1; i >= 0; i--)
        if (dp[i] == L)
            ans.push_back(v[i]), L--;
    reverse(ans.begin(), ans.end());
    for (auto i : ans) cout << i << " ";
}
```

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 [da8000]

```
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 [f34a85]

```
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 = --
             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 Removal Game [c4b594]

```cpp
// 兩個人比賽，每個人輪流取一個數字且只能是頭尾
// 問兩人都選得好，第一出手的人可取得的最大分數
void removalGame() {
    int n; cin >> n;
    vector<ll> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];
    vector<vector<ll>> dp(n, vector<ll>(n));
    // i 到 j 區間的最大 diff
    for (int i = n - 1; i >= 0; i--) {
        dp[i][i] = a[i];
        for (int j = i + 1; j < n; j++)
            dp[i][j] =
                max(a[i] - dp[i + 1][j], a[j] - dp[i][j - 1]);
    }
    // x + y = sum; // x - y = dp[0][n - 1]
    cout << (accumulate
        (a.begin(), a.end(), 0LL) + dp[0][n - 1]) / 2 << "\n";
}
```

9.7 Monotonic Queue [c9ba14]

```cpp
// 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";
}
```

9.8 SOS [7a4936]

```cpp
// 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<Z> dp(1 << m);
    for (int i = 0; i < n; i++)
        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[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].val()) - 1);
    }
    cout << ans << "\n";
}

// 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 - 與自己相同 - ~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] += 1;
        dp[a[i]][1] += 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";
    }
}
```

9.9 CHT [5f5c25]

```cpp
// 應用: dp(i) = h(i) + min/max(A(j)X(i) + B(j)), for j≤r(i)
// A(j), B(j) 可能包含 dp(j)，分別就是 m 跟 b
struct Line {
    ll m, b;
    Line(ll m = 0, ll b = 0) : m(m), b(b) {}
    ll eval(ll x) {
        return m * x + b;
    }
};
struct CHT { // 用在查詢單調斜率也單調
    int n, lptr, rptr;
    vector<Line> hull;
    CHT(int n_ = 0, Line init_ = Line()) {
        init(n_, init_);
    }
    void init(int n_ = 0, Line init_ = Line()) {
        n = n_; hull.resize(n); reset(init_);
    }
    void reset(Line init_ = Line()) {
        lptr = rptr = 0; hull[0] = init_;
    }
    bool pop_front(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 pop_back(Line &l1, Line &l2, Line &l3) {
        // 本題斜率遞減、上凸包
        // 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 insert(Line L) {
        while (rptr - lptr
            > 0 && pop_back(hull[rptr - 1], hull[rptr], L))
            rptr--;
        hull[++rptr] = L;
    }
    ll query(ll x) {
        while (rptr - lptr
            > 0 && pop_front(hull[lptr], hull[lptr + 1], x))
            lptr++;
        return hull[lptr].eval(x);
    }
};
```

9.10 DNC [49f715]

```cpp
// 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++) {
        // init dp[1][i]
    }
```

```
    for (int i = 2; i <= k; i++)
        rec(i, 1, n, 1, n);
    cout << dp[k][n] << "\n";
}
```

9.11 LiChao Segment Tree [588aa3]

```
// 應用: dp(i) = h(i) + min/max(A(j)X(i) + B(j)), for j ≤ r(i)
//         y    =  c   +          m  x   +   b
constexpr ll inf = 4E18;
struct Line {
    ll m, b;
    Line(ll m = 0, ll b = inf) : m(m), b(b) {}
    ll eval(ll x) const {
        return m * x + b;
    }
};
struct LiChaoSeg { // 取 max 再變換就好
    int n;
    vector<Line> info;
    LiChaoSeg(int n_ = 0) { init(n_); }
    void init(int n_) {
        n = n_;
        info.assign(4 << __lg(n), Line());
    }
    void update(Line line, int node, int l, int r) {
        int m = (l + r) / 2;
        bool left = line.eval(l) < info[node].eval(l);
        bool mid = line.eval(m) < info[node].eval(m);
        if (mid) swap(info[node], line); // 如果新線段比較好
        if (r - l == 1) return;
        else if (left != mid) update(line, 2 * node, l, m);
        // 代表左半有交點
        else update(line, 2 * node + 1, m, r);
        // 代表如果有交點一定在右半
    }
    void add_line(Line line) { update(line, 1, 0, n); }
    ll query(int x, int node, int l, int r) {
        if (r - l == 1) return info[node].eval(x);
        int m = (l + r) / 2;
        if (x < m) {
            return min(
                info[node].eval(x), query(x, 2 * node, l, m));
        } else {
            return min(info
                [node].eval(x), query(x, 2 * node + 1, m, r));
        }
    }
    ll query(int x) {
        return query(x, 1, 0, n);
    }
};
```

9.12 Codeforces Example [08fee8]

```
// CF 1932 pF
// 給你很多區間，你可以選一些點，重疊到的線段得到 1 分
// 請問在線段不重複的情況下，最多獲得幾分
void solve() {
    int n, m;
    cin >> n >> m;
    // 記錄每點有幾個線段
    // 再一個紀錄，包含這個點的左界
    vector<int> lside(n + 1, inf), cnt(n + 5, 0);
    for (int i = 0; i < m; i++) {
        int l, r; cin >> l >> r;
        lside[r] = min(lside[r], l);
        cnt[l]++;
        cnt[r + 1]--;
    }
    for (int i = 2; i <= n; i++)
        cnt[i] += cnt[i - 1];
    for (int i = n; i >= 2; i--)
        lside[i - 1] = min(lside[i - 1], lside[i]);
    vector<int> dp(n + 1);
    dp[0] = 0;
    for (int i = 1; i <= n; i++) {
        dp[i] = cnt[i];
        if (lside[i] != inf)
            dp[i] += dp[lside[i] - 1];
        dp[i] = max(dp[i], dp[i - 1]);
    }
    cout << dp[n] << "\n";
}

// CF 1935 pC
// 給你每個事件的 a, b，挑事件會把 a 全部加起來
// 再加上 max(bi) - min(bi)
void solve() {
    int n, k, ans = 0; cin >> n >> k;
    vector<pair<int, int>> v(n + 1);
    for (int i = 1; i <= n; i++) {
        int a, b; cin >> a >> b;
        v[i] = {a, b};
        if (a <= k) ans = 1;
    }
    sort(v.begin() +
         1, v.end(), [](pair<int, int> &a, pair<int, int> &b) {
        return a.second < b.second;
    }); // 用 bi 來排，考慮第 i 個時可以先扣
    vector<vector<int>> dp(n + 1, vector<int>(n + 1, inf));
    // 考慮 v[i] 時，選 j 個的 sum(ai) - min(bi)
    for (int i = 1; i <= n; i++) { // 滾動 dp
        for (int j = n; j >= 2; j--) {
            dp[i][j] = min
                (dp[i - 1][j], dp[i - 1][j - 1] + v[i].first);
            // min(不選, 選)
            if (dp[i
                - 1][j - 1] + v[i].first + v[i].second <= k) {
                // 假如可以選，更新 ans 時再加回去 bi
                ans = max(ans, j);
            }
        }
        dp[i][1] = min(dp[i - 1][1], v[i].first - v[i].second);
    }
    cout << ans << "\n";
}
```

10 Geometry

10.1 Basic [d41d8c]

```
template<class T>
struct Point {
    T x, y;
    Point(const T &x_ = 0, const T &y_ = 0) : x(x_), y(y_) {}
    template<class U>
    operator Point<U>() {
        return Point<U>(U(x), U(y));
    }
    Point &operator+=(const Point &p) & {
        x += p.x; y += p.y; return *this;
    }
    Point &operator-=(const Point &p) & {
        x -= p.x; y -= p.y; return *this;
    }
    Point &operator*=(const T &v) & {
        x *= v; y *= v; return *this;
    }
    Point &operator/=(const T &v) & {
        x /= v; y /= v; return *this;
    }
    Point operator-() const {
        return Point(-x, -y);
    }
    friend Point operator+(Point a, const Point &b) {
        return a += b;
    }
    friend Point operator-(Point a, const Point &b) {
        return a -= b;
    }
    friend Point operator*(Point a, const T &b) {
        return a *= b;
    }
    friend Point operator/(Point a, const T &b) {
        return a /= b;
    }
    friend Point operator*(const T &a, Point b) {
        return b *= a;
    }
    friend bool operator==(const Point &a, const Point &b) {
        return a.x == b.x && a.y == b.y;
    }
    friend istream &operator>>(istream &is, Point &p) {
        return is >> p.x >> p.y;
    }
    friend ostream &operator<<(ostream &os, const Point &p) {
        return os << "(" << p.x << ", " << p.y << ")";
    }
};
template<class T>
T dot(const Point<T> &a, const Point<T> &b) {
    return a.x * b.x + a.y * b.y;
}
template<class T>
T cross(const Point<T> &a, const Point<T> &b) {
    return a.x * b.y - a.y * b.x;
}
template<class T>
T square(const Point<T> &p) {
    return dot(p, p);
}
template<class T>
double length(const Point<T> &p) {
    return sqrt(double(square(p)));
}
template<class T>
Point<T> normalize(const Point<T> &p) {
    return p / length(p);
}
template<class T>
Point<T> rotate(const Point<T> &a) {
    return Point(-a.y, a.x);
}
template<class T>
int sgn(const Point<T> &a) {
    return a.y > 0 || (a.y == 0 && a.x > 0) ? 1 : -1;
}

template<class T>
struct Line {
```

```
    Point<T> a;
    Point<T> b;
    Line(const Point<T> &a_ = Point<T>()
         , const Point<T> &b_ = Point<T>()) : a(a_), b(b_) {}
};
template<class T>
double length(const Line<T> &l) {
    return length(l.a - l.b);
}
template<class T>
bool parallel(const Line<T> &l1, const Line<T> &l2) {
    return cross(l1.b - l1.a, l2.b - l2.a) == 0;
}
template<class T>
double distance(const Point<T> &a, const Point<T> &b) {
    return length(a - b);
}
template<class T>
double distancePL(const Point<T> &p, const Line<T> &l) {
    return abs(cross(l.a - l.b, l.a - p)) / length(l);
}
template<class T>
double distancePS(const Point<T> &p, const Line<T> &l) {
    if (dot(p - l.a, l.b - l.a) < 0)
        return distance(p, l.a);
    if (dot(p - l.b, l.a - l.b) < 0)
        return distance(p, l.b);
    return distancePL(p, l);
}
template<class T>
bool pointOnLineLeft(const Point<T> &p, const Line<T> &l) {
    return cross(l.b - l.a, p - l.a) > 0;
}
template<class T>
Point<T
    > lineIntersection(const Line<T> &l1, const Line<T> &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));
}
template<class T>
bool pointOnSegment(const Point<T> &p, const Line<T> &l) {
    return cross(p - l.a, l.b - l.a) == 0 &&
          min(l.a.x, l.b.x) <= p.x && p.x <= max(l.a.x, l.b.x)
        && min
             (l.a.y, l.b.y) <= p.y && p.y <= max(l.a.y, l.b.y);
}
template<class T>
bool pointInPolygon
    (const Point<T> &a, const vector<Point<T>> &p) {
    int n = p.size(), t = 0;
    for (int i = 0; i < n; i++)
        if (pointOnSegment(a, Line(p[i], p[(i + 1) % n])))
            return true;
    for (int i = 0; i < n; i++) {
        auto u = p[i];
        auto v = p[(i + 1) % n];
        if (u.x < a.
             x && v.x >= a.x && pointOnLineLeft(a, Line(v, u)))
            t ^= 1;
        if (u.x >= a
             .x && v.x < a.x && pointOnLineLeft(a, Line(u, v)))
            t ^= 1;
    }
    return t == 1;
}
// 0 : strictly outside
// 1 : on boundary
// 2 : strictly inside
template<class T>
int pointInConvexPolygon
    (const Point<T> &a, const vector<Point<T>> &p) {
    int n = p.size();
    if (n == 0) {
        return 0;
    } else if (n <= 2) {
        return pointOnSegment(a, Line(p[0], p.back()));
    }
    if (pointOnSegment(a, Line(p[0],
          p[1])) || pointOnSegment(a, Line(p[0], p[n - 1]))) {
        return 1;
    } else if (pointOnLineLeft(a, Line(p[1],
          p[0])) || pointOnLineLeft(a, Line(p[0], p[n - 1]))) {
        return 0;
    }
    int lo = 1, hi = n - 2;
    while (lo < hi) {
        int x = (lo + hi + 1) / 2;
        if (pointOnLineLeft(a, Line(p[0], p[x]))) {
            lo = x;
        } else {
            hi = x - 1;
        }
    }
    if (pointOnLineLeft(a, Line(p[lo], p[lo + 1]))) {
        return 2;
    } else {
        return pointOnSegment(a, Line(p[lo], p[lo + 1]));
    }
}
template<class T>
bool lineIntersectsPolygon
    (const Line<T> &l, const vector<Point<T>> &p) {
    int n = p.size();
    Point<T> a = l.a, b = l.b;
    for (int i = 0; i < n; i++) {
        Line<T> 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;
}
// 0 : not intersect
// 1 : strictly intersect
// 2 : overlap
// 3 : intersect at endpoint
template<class T>
tuple<int, Point<T>, Point<T>> segmentIntersection
    (const Line<T> &l1, const Line<T> &l2) {
    if (max(l1.a.x, l1.b.x) < min(l2.a.x, l2.b.x))
        return {0, Point<T>(), Point<T>()};
    if (min(l1.a.x, l1.b.x) > max(l2.a.x, l2.b.x))
        return {0, Point<T>(), Point<T>()};
    if (max(l1.a.y, l1.b.y) < min(l2.a.y, l2.b.y))
        return {0, Point<T>(), Point<T>()};
    if (min(l1.a.y, l1.b.y) > max(l2.a.y, l2.b.y))
        return {0, Point<T>(), Point<T>()};
    if (cross(l1.b - l1.a, l2.b - l2.a) == 0) {
        if (cross(l1.b - l1.a, l2.a - l1.a) != 0) {
            return {0, Point<T>(), Point<T>()};
        } 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);
            Point<T> p1(max(minx1, minx2), max(miny1, miny2));
            Point<T> 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, Point<T>(), Point<T>()};
    Point p = lineIntersection(l1, l2);
    if (cp1 != 0 && cp2 != 0 && cp3 != 0 && cp4 != 0) {
        return {1, p, p};
    } else {
        return {3, p, p};
    }
}
template<class T>
double distanceSS(const Line<T> &l1, const Line<T> &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)});
}
template<class T>
bool segmentInPolygon
    (const Line<T> &l, const vector<Point<T>> &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, Line(u, v));
        if (t == 1) return false;
        if (t == 0) continue;
        if (t == 2) {
            if (pointOnSegment(v, l) && v != l.a && v != l.b)
                if (cross(v - u, w - v) > 0)
                    return false;
        } else {
            if (p1 != u && p1 != v) {
                if (pointOnLineLeft(l.a, Line(v, u))
                    || pointOnLineLeft(l.b, Line(v, u)))
                    return false;
            } else if (p1 == v) {
                if (l.a == v) {
                    if (pointOnLineLeft(u, l)) {
                        if (pointOnLineLeft(w, l)
                            && pointOnLineLeft(w, Line(u, v)))
                            return false;
                    } else {
                        if (pointOnLineLeft(w, l)
```

```
                            || pointOnLineLeft(w, Line(u, v)))
                            return false;
                    }
                } else if (l.b == v) {
                    if (pointOnLineLeft(u, Line(l.b, l.a))) {
                        if (pointOnLineLeft(w, Line(l.b, l.a))
                            && pointOnLineLeft(w, Line(u, v)))
                            return false;
                    } else {
                        if (pointOnLineLeft(w, Line(l.b, l.a))
                            || pointOnLineLeft(w, Line(u, v)))
                            return false;
                    }
                } else {
                    if (pointOnLineLeft(u, l)) {
                        if (pointOnLineLeft(w, Line(l.b, l.a))
                            || pointOnLineLeft(w, Line(u, v)))
                            return false;
                    } else {
                        if (pointOnLineLeft(w, l)
                            || pointOnLineLeft(w, Line(u, v)))
                            return false;
                    }
                }
            }
        }
    }
    return true;
}
template<class T>
vector<Point<T>> convexHull(vector<Point<T>> a) {
    sort(a.begin()
         , a.end(), [](const Point<T> &l, const Point<T> &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<Point<T>> h(a.size() + 1);
    int s = 0, t = 0;
    for (int i = 0; i < 2; i++, s = --t) {
        for (Point<T> 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};
}
template<class T>
vector<Point<T>> hp(vector<Line<T>> lines) {
    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<T>> ls;
    deque<Point<T>> ps;
    for (auto l : lines) {
        if (ls.empty()) {
            ls.push_back(l);
            continue;
        }
        while (!ps.empty() && !pointOnLineLeft(ps.back(), l))
            ps.pop_back(), ls.pop_back();
        while (!ps.empty() && !pointOnLineLeft(ps[0], l))
            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 (!pointOnLineLeft(ls.back().a, l)) {
                    assert(ls.size() == 1);
                    ls[0] = l;
                }
                continue;
            }
            return {};
        }
        ps.push_back(lineIntersection(ls.back(), l));
        ls.push_back(l);
    }
    while (!ps.empty() && !pointOnLineLeft(ps.back(), ls[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());
}
using P = Point<ll>;
```

10.2 Min Euclidean Distance [478e73]

```
void minEuclideanDistance() {
    int n; cin >> n;
    constexpr ll inf = 8E18;
    vector<Point<ll>> a(n);
    for (int i = 0; i < n; i++) {
        ll x, y;
        cin >> x >> y;
        a[i] = Point<ll>(x, y);
    }
    struct sortY {
        bool operator
            ()(const Point<ll> &a, const Point<ll> &b) const {
            return a.y < b.y;
        }
    };
    struct sortXY {
        bool operator
            ()(const Point<ll> &a, const Point<ll> &b) const {
            return a.x == b.x ? a.y < b.y : a.x < b.x;
        }
    };
    sort(a.begin(), a.end(), sortXY());
    vector<Point<ll>> t(n);
    auto devide = [&](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 << devide(devide, 0, n - 1) << "\n";
}
```

10.3 Max Euclidean Distance [4aa1f0]

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

10.4 Lattice Points [46d224]

```
void latticePoints() {
    // Area 求法與 Polygun 內整數點數
    int n; cin >> n;
    vector<Point<ll>> 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<Point<ll>>& 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 += std::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 [9380bf]

```
template<class T>
pair<T, Point<T>> minCircleCover(vector<Point<T>> &a) {
    random_shuffle(a.begin(), a.end());
    int n = a.size();
    Point<T> c = a[0]; T r = 0;
    for (int i = 1; i < n; i++) {
        if (T(length(c - a[i]) - r) > 0.0) {
            c = a[i], r = 0;
            for (int j = 0; j < i; j++) {
                if (T(length(c - a[j]) - r) > 0.0) {
                    c = (a[i] + a[j]) / 2.0;
```

```
                    r = length(c - a[i]);
                    for (int k = 0; k < j; k++) {
                        if (T(length(c - a[k]) - r) > 0.0) {
                            Point<T> p = (a[j] + a[i]) / 2;
                            Point<T> q = (a[j] + a[k]) / 2;
                            if (cross(a[j] - a[i],
                                  a[k] - a[j]) == 0) continue;
                            c = lineIntersection(Line(p,
                                 p + rotate(a[j] - a[i])), Line
                                 (q, q + rotate(a[k] - a[j])));
                            r = length(c - a[i]);
                        }
                    }
                }
            }
        }
    }
    return {r, c};
}
```

10.6 Min Rectangle Cover [8bd345]

```
template<class T>
pair<T,
     vector<Point<T>>> minRectangleCover(vector<Point<T>> a) {
    if (a.size() <= 2) return {0, {}};
    auto get = [&](const Point<T> &p, const Line<T> &l) -> T {
        return abs(cross(l.a - l.b, l.a - p).x);
    };
    int n = a.size(), j = 2, l = 1, r = 1;
    a.push_back(a.front());
    D th, tw, area = numeric_limits<double>::infinity();
    vector<Point<T>> ans;
    for (int i = 0; i < n; i++) {
        while (get(a[j], Line(a[i], a[i +
             1])) <= get(a[(j + 1) % n], Line(a[i], a[i + 1])))
            j = (j + 1) % n;
        while (dot(a[i + 1] - a[i], a[r] - a[i])
            <= dot(a[i + 1] - a[i], a[(r + 1) % n] - a[i]))
            r = (r + 1) % n;
        if (i == 0) l = j;
        while (dot(a[i + 1] - a[i], a[l] - a[i])
            >= dot(a[i + 1] - a[i], a[(l + 1) % n] - a[i]))
            l = (l + 1) % n;
        D th = get(a[j], Line(a[i], a[i + 1]));
        D tw = dot(a[i] - a[i + 1],
              a[l] - a[i]) + dot(a[i + 1] - a[i], a[r] - a[i]);
        if (th * tw / square(a[i + 1] - a[i]) < area) {
            ans.clear
                 (), area = th * tw / square(a[i + 1] - a[i]);
            Line l1(a[i], a[i + 1]);
            for (auto p : {a[r], a[j], a[l], a[i]}) {
                Line l2 = Line(p, p + rotate(l1.a - l1.b));
                if (cross(l1.a - l1.b, p - l1.a) == 0) {
                    ans.push_back(p);
                    l1 = Line(p, p + rotate(l1.a - l1.b));
                } else {
                    Point<T> res = lineIntersection(l1, l2);
                    ans.push_back(res);
                    l1.a = res, l1.b = p;
                }
            }
        }
    }
    return {area, ans};
}
```

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 [4cf2c8]

```
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 [fa7d62]

```
from decimal import * # 無誤差浮點數
from fractions import * # 分數
from random import *
from math import *
# set decimal prec 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(x)
print(a, b, c)
print(*arr)
# set
S = set(); S.add((a, b)); S.remove((a, b))
if not (a, b) in S:
# 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 [70f2dd]

```
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;
    }
};
```

12.3 Multiple [79b47c]

```
// 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> small_mul(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 ? small_mul(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 [1169e0]

```
private:
    vector<int> small_div(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 = small_div(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;
    }
```

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)
```