Contents

1 Basic

1.1 install vscode [d41d8c]

```
// 如何安裝 vscode
// 1. 下載 vscode & msys2
// 2. 在跳出的 terminal 中 / 或打開 ucrt64，打上
     "pacman -S --needed base-devel mingw-w64-x86_64-toolchain"
// 3. 環境變數加上 C:\\msys64\\ucrt64\\bin
// 4. 重開 vscode, 載 C/C++, 運行, 編譯器選擇 g++
// 5. 打開 settings -> compiler -> add compilerPath
      -> 在 "" 裡打上 C:\\msys64\\ucrt64\\bin\\g++.exe
```

1.2 default code [3cd57c]

```
#include <bits/stdc++.h>
#define all(x) (x).begin(), (x).end()
#define pii pair<int, int>
using namespace std;
using ll = long long;
const int mod = 1e9 + 7;

void solve() {

}

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

1.3 compare fuction [4bc3e0]

```
struct cmp {    // 在有 template 的資結使用
    bool operator()(const int &a, const int &b) const {
        return a < b;
    }
// sort, bound 不用 struct
// priority queue 小到大是 > , set 是 <
// set 不能 = , multiset 要 =
// 每個元素都要比到，不然會不見
// pbds_multiset 不要用 lower_bound
// 如果要 find, 插入 inf 後使用 upper_bound
// 內建 multiset 可以跟 set 一樣正常使用
// 如果有自定義比較結構就比照以上
};

struct cmp {    // 要在 template 的資結用外部變數
    vector<int> &v;
    cmp(vector<int>& vec) : v(vec) {}
    bool operator() (int a, int b) const {
        return v[a] > v[b];
    }
// main: cmp cmp1(vector);
// priority_queue<int, vector<int>, cmp> pq(cmp1);
};
```

1.4 pbds [e28ae8]

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

1.5 浮點數誤差 [a0d4e5]

```
struct EDouble {
    double x;
    constexpr static double Eps = 1e-9;
    constexpr EDouble() : x{} {}
    constexpr EDouble(double v) : x{v} {}
    constexpr double val() const {
        return x;
    }
    explicit constexpr operator double() const {
        return x;
    }
    constexpr EDouble operator-() const {
        return EDouble(-x);
    }
    constexpr EDouble &operator+=(const EDouble &rhs) & {
        x += rhs.x;
        return *this;
    }
    constexpr EDouble &operator-=(const EDouble &rhs) & {
        x -= rhs.x;
        return *this;
    }
    constexpr EDouble &operator*=(const EDouble &rhs) & {
        x *= rhs.x;
        return *this;
    }
    constexpr EDouble &operator/=(const EDouble &rhs) & {
        assert(fabs(rhs.x) > Eps);
        x /= rhs.x;
        return *this;
    }
    friend constexpr
          EDouble operator+(EDouble lhs, const EDouble &rhs) {
        lhs += rhs;
        return lhs;
    }
    friend constexpr
          EDouble operator-(EDouble lhs, const EDouble &rhs) {
        lhs -= rhs;
        return lhs;
    }
    friend constexpr
          EDouble operator*(EDouble lhs, const EDouble &rhs) {
        lhs *= rhs;
        return lhs;
    }
    friend constexpr
          EDouble operator/(EDouble lhs, const EDouble &rhs) {
        lhs /= rhs;
        return lhs;
    }
    friend constexpr bool
          operator<(const EDouble &lhs, const EDouble &rhs) {
        return lhs.x - rhs.x < -Eps;
    }
    friend constexpr bool
          operator>(const EDouble &lhs, const EDouble &rhs) {
        return lhs.x - rhs.x > Eps;
    }
    friend constexpr bool
          operator==(const EDouble &lhs, const EDouble &rhs) {
        return fabs(lhs.x - rhs.x) < Eps;
    }
    friend constexpr bool
          operator<=(const EDouble &lhs, const EDouble &rhs) {
        return lhs < rhs || lhs == rhs;
    }
    friend constexpr bool
          operator>=(const EDouble &lhs, const EDouble &rhs) {
        return lhs > rhs || lhs == rhs;
    }
    friend constexpr bool
          operator!=(const EDouble &lhs, const EDouble &rhs) {
        return !(lhs == rhs);
    }
    friend istream &operator>>(istream &is, EDouble &a) {
        double v; is >> v;
        a = EDouble(v);
        return is;
    }
    friend ostream &operator<<(ostream &os, const EDouble &a) {
        return os << fixed << setprecision(7) << a.val
             () + (a.val() > 0 ? Eps : a.val() < 0 ? -Eps : 0);
    } // Eps should < precision
};
namespace std {
    template<>
    class numeric_limits<EDouble> {
    public:
        static constexpr EDouble max() noexcept {
            return EDouble(numeric_limits<double>::max());
        }
        static constexpr EDouble min() noexcept {
            return EDouble(numeric_limits<double>::min());
        }
    };
}
using E = EDouble;
```

2 Graph

2.1 DFS 跟 BFS [cdd1d5]

```
int main() {
    int n;
    vector<vector<int>> adj(n + 1, vector<int>());
    // dfs_graph
    vector<bool> vis(n + 1, 0);
    auto dfs = [&](auto self, int u) -> void {
        if (vis[u]) return;
        vis[u] = true;
        for (auto v: adj[u]) {
            self(self, v);
        }
    };
    dfs(dfs, 1);
    // bfs
    vector<int> depth(n + 1, 1e9);
    queue<int> q;
    auto bfs = [&](auto self, int u) -> void {
        vis[u] = true;
        depth[u] = 0;
        q.push(u);
        while (!q.empty()) {
            int u = q.front(); q.pop();
            for (auto v : adj[u]) {
                if (vis[v]) continue;
                vis[v] = true;
                depth[v] = depth[u] + 1;
                q.push(v);
            }
        }
    };
    bfs(bfs, 1);
}
```

2.2 Prim [f00ec0]

```
auto prim =
      [&](int n, vector<vector<pair<int, int>>> &adj) -> bool {
    int node_sz = 0;
    priority_queue<pair<int, int>,
          vector<pair<int, int>>, greater<pair<int, int>>> pq;
    pq.push({0, 1}); // w, vertex
    vector<bool> vis(n);
    while (!pq.empty()) {
        auto [u, w] = pq.top(); pq.pop();
        if (vis[u]) continue;
        vis[u] = true;
        node_sz++;
        for (auto v : adj[u]) {
            if (!vis[v.first]) {
                pq.push({v.second, v.first});
            }
        }
    }
    if (node_sz == n) return true;
    return false;
};
```

2.3 BellmanFord [02f480]

```
// 用 Bellman Ford 找負環
vector<array<int, 3>> graph;    // u, v, w
int main() {
    int src = 0;
    int n, m;    cin >> n >> m;
    vector<int> par(n + 1), dis(n + 1, 1e9);
    for (int i = 0; i < m; i++) {
        int a, b, w; cin >> a >> b >> w;
        graph.push_back({a, b, w});
    }
    dis[1] = 0;
    for (int i = 0; i <= n; i++) {
        src = 0;
        for (auto [u, v, w] : graph) {
            if (dis[v] > dis[u] + w) {
                dis[v] = dis[u] + w;
                par[v] = u;
                src = v;
            }
        }
    }
    if (src) {  // 到第 n + 1 次還在鬆弛
        vector<int> ans;
        cout << "YES" << endl;
        for (int
              i = 0; i <= n; i++) src = par[src]; // 找那個負環
        ans.push_back(src);
        for (int
             i = par[src]; i != src; i = par[i]) {  // 輸出負環
            ans.push_back(i);
        }
        ans.push_back(src);
        reverse(ans.begin(), ans.end());
        for (auto i : ans) {
            cout << i << " ";
        }
    }
    else {
        cout << "NO" << "\n";
    }
}
```

2.4 負權最大距離 [2148ca]

```
// CSES High Score
void dfs(int u, vector<int> &vis, vector<vector<int>> &adj) {
    if (vis[u]) return;
    vis[u] = 1;
    for (int v : adj[u]) {
        dfs(v, vis, adj);
    }
}
signed main() {
    int n, m; cin >> n >> m;
    vector<array<int, 3>> edges;
    vector<vector<int>> adj(n + 1);
    vector<int> dis(n + 1), vis(n + 1);
    while (m--) {
        int u, v, w;
        cin >> u >> v >> w;
        edges.push_back({u, v, w});
        adj[u].push_back(v);
    }
    fill(dis.begin(), dis.end(), -1e18);
    dis[1] = 0;
    for (int i = 1; i <= n; i++) {
        for (auto [u, v, w] : edges) {
            if (dis[u] != -1e18 && dis[v] < dis[u] + w) {
                dis[v] = dis[u] + w;
                if (i == n) {
                    dfs(v, vis, adj);
                }
            }
        }
    }
    if (vis[n]) cout << -1;
    else cout << dis[n];
}
```

2.5 FloydWarshall [206b76]

```
const int inf = 1e18;
int main() {
    int n, m, q; cin >> n >> m >> q;
    vector<vector<int>> graph(n + 1, vector<int>(n + 1, inf));
    vector<vector<int>> dis(n + 1, vector<int>(n + 1));
    for (int i = 0; i < m; i++) {
        int u, v, w; cin >> u >> v >> w;
        cin >> u >> v >> w;
        graph[u][v] = min(graph[u][v], w);
        graph[v][u] = min(graph[v][u], w);
    }
    for (int i = 0; i <= n; i++) {
        for(int j = 0; j <= n; j++) {
            dis[i][j] = graph[i][j];
        }
    }
    for (int i = 0; i <= n; i++) // 自己到自己是 0
        dis[i][i] = 0;

    for (int k = 1; k <= n; k++) {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                dis[i][j
                     ] = min(dis[i][j], dis[i][k] + dis[k][j]);
            }
        }
    }
    for (int i = 0; i < q; i++) {
        int u, v; cin >> u >> v;
        cout << (dis[u][v] >= inf ? -1 : dis[u][v]) << "\n";
    }
}
```

2.6 歐拉環與歐拉路 [0911ed]

```cpp
// 無向圖、尤拉環: 檢查每個點的出度為偶數
// 有向圖、
    尤拉路: 可以看成 1 走到 n，所以檢查所有點的出度等於入度
int n, m;
const int maxn = 1e5 + 5;
vector<set<int>> adj;
vector<int> in;
void dfs(int now, vector<int> &road) {
    while (!adj[now].empty()) {
        int nxt = *adj[now].begin();
        adj[now].erase(nxt);
        dfs(nxt, road);
    }
    road.push_back(now);
}
void solve() {
    cin >> n >> m;
    in.assign(n + 1, 0);
    adj.assign(n + 1, set<int>());
    for (int i = 1; i <= m; i++) {
        int u, v; cin >> u >> v;
        adj[u].insert(v);
        in[v]++;
    }
    in[1]++;
    in[n]--;
    for (int i = 1; i <= n; i++) {
        if(adj[i].size() != in[i]) {
            cout << "IMPOSSIBLE";
            return;
        }
    }
    vector<int> road;
    dfs(1, road);
    if (road.size() != m + 1) {
        cout << "IMPOSSIBLE";
        return;
    }
    reverse(road.begin(), road.end());
    for(auto i : road) cout << i << " ";
}
```

2.7 SCC [5d3e16]

```cpp
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;
        vector<int> 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 [170604]

```cpp
struct VBCC {
    int n, cur;
    vector<vector<int>> adj;
    vector<int> dfn, low, parent;
    vector<bool> is_cut;
    VBCC(int n_ = 0) {
        init(n_);
    }
    void init(int n_) {
        n = n_;
        adj.assign(n, {});
        dfn.assign(n, -1);
        low.resize(n);
        parent.assign(n, -1);
        is_cut.assign(n, false);
        cur = 0;
    }
    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    void dfs(int x) {
        int children = 0;
        dfn[x] = low[x] = cur++;
        for (int v : adj[x]) {
            if (dfn[v] == -1) {
                children++;
                parent[v] = x;
                dfs(v);
                low[x] = min(low[x], low[v]);

                if (parent[x] != -1 && low[v] >= dfn[x]) {
                    is_cut[x] = true;
                }
            } else if (v != parent[x]) {
                low[x] = min(low[x], dfn[v]);
            }
        }
        if (parent[x] == -1 && children > 1) {
            is_cut[x] = true;
        }
    }
    void work() {
        for (int i = 0; i < n; i++) {
            if (dfn[i] == -1) {
                dfs(i);
            }
        }
    }
};
```

2.9 EBCC [49d862]

```cpp
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_) {
        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; // BCC 內節點數
        vector<int> cnte; // BCC 內邊數
    };
    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 [eeddc1]

```
// CSES Giant Pizza
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);
    }
    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; }
};
int main() {
    int m, n; cin >> m >> n;
    TwoSat ts(n);
    for (int i = 0; i < m; ++i) {
        int u, v; char x, y;
        cin >> x >> u >> y >> v;
            ts.addClause(u - 1, x == '+', v - 1, y == '+');
    }
    if (ts.satisfiable()) {
        for (int i = 0; i < n; ++i) {
            cout << (ts.answer()[i] ? '+' : '-') << " ";
        }
    }
    else cout << "IMPOSSIBLE\n";
}
```

2.11 Planets Cycles [71ac0e]

```
vector<int> dis, v;
vector<bool> vis;
int step;
queue<int> path;
void dfs(int x) {
    path.push(x);
    if (vis[x]) {
        step += dis[x];
        return;
    }
    vis[x] = true;
    step++;
    dfs(v[x]);
}
// count path_dis to rep
int main() {
    int n; cin >> n;
    v.assign(n + 1, 0);
    dis.assign(n + 1, 0);
    vis.assign(n + 1, false);
    for (int i = 1; i <= n; i++) {
        cin >> v[i];
    }
    for (int i = 1; i <= n; i++) {
        step = 0;
        int is_outof_cycle = 1;
        dfs(i);
        while (!path.empty()) {
            if (path.front() == path.back()) {
                is_outof_cycle = 0;
            }
            dis[path.front()] = step;
            step -= is_outof_cycle;
            path.pop();
        }
    }
    for (int i = 1; i <= n; i++) {
        cout << dis[i] << ' ';
    }
    cout << '\n';
}
```

2.12 Planet Queries II [872f72]

```
// 在有向圖中，從 A 到 B 的最短距離
// 保證出度是 1 所以對 1 個點來說，從他出發只可能遇到一個環
int n, q;
int dp[200005][30];   // 倍增表
vector<vector<int>> cycles;
vector<int
     > no, cycle_idx, vis; // Order & Can be in cycle, or out
void set_out_of_cycle_no(int now, unordered_set<int> &done) {
    // 把不在環內的也編號，v 是 u 的編號 -1
    if (done.find(now) != done.end()) return;
    set_out_of_cycle_no(dp[now][0], done);
    done.insert(now);   // post order
    no[now] = no[dp[now][0]] - 1;
}
int wiint_go_to(int u, int k) { // 回傳當 u 走 k 步時會到的地方
    for (int i = 0; i <= 18; i++) {
        if (k & (1 << i)) {
            u = dp[u][i];
        }
    }
    return u;
}
void find_cycle(int now) {
    unordered_set<int> appear;
    vector<int> v;
    bool flag = true;   // 代表有環
    while (appear.find(now) == appear.end()) {
        appear.insert(now);
        v.push_back(now);
        if (vis[now]) {
            flag = false;
            break;
        }
        now = dp[now][0];
    }
    for (auto i : v)  vis[i] = true;
    if (!flag)  return;
    // now 是環的起點，我們先找到他在 v 的哪裡
    int z = find(v.begin(), v.end(), now) - v.begin();
    vector<int> cycle(v.begin() + z, v.end());
    cycles.push_back(cycle);
}
int main() {
    cin >> n >> q;
    no.assign(n + 1, -1);
    cycle_idx.assign(n + 1, -1);
    vis.assign(n + 1, 0);
    for (int u = 1; u <= n; u++) cin >> dp[u][0];
    for (int i = 1; i <= 18; i++) // 倍增表
        for (int u = 1; u <= n; u++)
            dp[u][i] = dp[dp[u][i - 1]][i - 1];
    for (int i = 1; i <= n; i++) {
        if (!vis[i]) find_cycle(i);
    }
    int idx = 0;
    unordered_set<int> done;
    for (auto &i : cycles) {
        int c = 0;
        for (auto &j : i) {
            no[j] = c++;
            cycle_idx[j] = idx;
            done.insert(j);
        }
        idx++;
    }
    for (int i = 1; i <= n; i++) set_out_of_cycle_no(i, done);
    for (int i = 1; i <= q; i++) {
        int u, v; cin >> u >> v;
        // 在同個環內
        if (cycle_idx[u] == cycle_idx
             [v] && cycle_idx[u] != -1 && cycle_idx[v] != -1) {
            int cyc_size = cycles[cycle_idx[u]].size();
            cout <<
                 (no[v] - no[u] + cyc_size) % cyc_size << "\n";
        }
        // 都不再環內
        else if (cycle_idx[u] == -1 &&
              cycle_idx[v] == -1) { // Both are not in a Cycle
            if (no[u] > no[v]) {
                cout << -1 << "\n";
                continue;
            }
            if (wiint_go_to(u, no[v] - no[u]) == v) {
                cout << no[v] - no[u] << "\n";
            }
            else cout << -1 << "\n";
        }
        else if (cycle_idx[u]
             == -1 && cycle_idx[v] != -1) { // v 在環內，二分搜
            int l = -1, r = n;
            while (l <= r) {
                int m = (l + r) / 2;
                if (cycle_idx[wiint_go_to
                     (u, m)] == cycle_idx[v]) r = m - 1;
                else l = m + 1;
            }
            if (l <= n) {   // 如果 n 步內可以到
                int in_cycle_of_u = wiint_go_to(u, l);
                int cycle_size = cycles[cycle_idx[v]].size();
                cout << l + (no[v] - no[in_cycle_of_u
                     ] + cycle_size) % cycle_size << "\n";
            }
            else cout << -1 << "\n";
        }
        else { // u 在環內 b 不在，直接不可能
            cout << -1 << "\n";
        }
    }
}
```

3 Data Structure

3.1 BIT [d41d8c]

```
template <typename 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) { // 找到最小的 x, 使得 sum(x) > k
        int x = 0;
        T cur{};
        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);
    }
};
```

3.2 RangeBit [d41d8c]

```
template <typename 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) { // 找到最小的 x, 使得 sum(x) > k
        int x = 0;
        T cur{};
        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);
    }
};
```

3.3 DSU [d41d8c]

```
struct DSU {
    int n;
    vector<int> boss, siz;
    DSU() {}
    DSU(int n_) {
        init(n_);
    }
    void init(int n_) {
        n = n_;
        boss.resize(n);
        iota(boss.begin(), boss.end(), 0);
        siz.assign(n, 1);
    }
    int find_boss(int x) {
        if (boss[x] == x) return x;
        return boss[x] = find_boss(boss[x]);
    }
    bool same(int x, int y) {
        return find_boss(x) == find_boss(y);
    }
    bool merge(int x, int y) {
        x = find_boss(x);
        y = find_boss(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_boss(x)];
    }
};

struct DSU {
    int n;
    vector<int> boss, siz, stk;
    DSU() {}
    DSU(int n_) {
        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)];
    }
};
```

3.4 線段樹 [d41d8c]

```
template <class Info>
struct Seg {    // 左閉右開寫法
    int n;
    vector<Info> info;
    Seg() : n(0) {}
    Seg(int n_, Info v_ = Info()) {
        init(n_, v_);
    }
    template <class T>
    Seg(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());
        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 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;
        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;
        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); }
    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;
        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);
    }
};
// ---define structure and info plus---
struct Info {
    int n = 1;
    int sum = 0;
};
Info operator+(const Info &a, const Info &b) {
    return { a.n + b.n, a.sum + b.sum };
}
```

3.5 懶標線段樹 [d41d8c]

```
template <class Info, class Tag>
struct LazySeg {    // 左閉右開寫法
    int n;
    vector<Info> info;
    vector<Tag> tag;
    LazySeg() : n(0) {}
    LazySeg(int n_, Info v_ = Info()) {
        init(n_, v_);
    }
    template <class T>
    LazySeg(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 range_apply
     (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);
        range_apply(p * 2, l, m, ql, qr, v);
        range_apply(p * 2 + 1, m, r, ql, qr, v);
        pull(p);
    }
    void range_apply(int l, int r, const Tag &v) {
        range_apply(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);
        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);
    }
};
// ---define structure and info plus---
struct Tag { // 有些 Tag 不用 push 例如 sweepLine
    int set_val; int add;
    void apply(const Tag& v) {
        if (v.set_val) {
            set_val = v.set_val;
            add = v.add;
        }
        else {
            add += v.add;
        }
    }
};
struct Info {
    int sum;
    void apply(int l, int r, const Tag &v) {
        if (v.set_val) {
            sum = (r - l) * v.set_val;
        }
        sum += (r - l) * v.add;
    }
    // Info& operator=(const Info &rhs) {
    //     // 部分 assignment 使用
    //     return *this;
    // }
};
Info operator+(const Info &a, const Info &b) {
    return { a.sum + b.sum };
}
```

3.6 莫隊 [d41d8c]

```
struct query {
    int l, r, id;
} typedef query;
void MO(int n, vector<query> &queries) {
    int block = sqrt(n);
    function <bool(query, query)> cmp = [&](query a, query b) {
        int block_a = a.l / block;
        int block_b = b.l / block;
        if (block_a != block_b) return block_a < block_b;
        return a.r < b.r;
    };
    sort(queries.begin(), queries.end(), cmp);
}
void compress(vector<int> &nums) {
    vector<int> sorted = nums;
    sort(sorted.begin(), sorted.end());
    sorted.erase
     (unique(sorted.begin(), sorted.end()), sorted.end());
    for (int i = 0; i < nums.size(); i++) {
        nums[i] = lower_bound(sorted.begin
         (), sorted.end(), nums[i]) - sorted.begin() + 1;
    }
}
```

3.7 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);
}
```

4 Flow

4.1 Dinic [287fe8]

```
template<class T>
struct Dinic {
    struct Edge {
        int to;
        T flow, cap; // 流量跟容量
    };
    int n, m, s, t;
    T INF_FlOW = numeric_limits<T>::max() / 2;
    vector<vector<int>> adj; // 此點對應的 edges 編號
    vector<Edge> edges; // 幫每個 edge 編號
    vector<int> dis, ptr;
    Dinic(int n_ = 0) { init(n_); }
    void init(int n_ = 0) {
        n = n_; m = 0;
        dis.resize(n); ptr.resize(n);
        adj.assign(n, vector<int>{});
        edges.clear();
    }
    void add_edge(int u, int v, T cap) {
        // 偶數 id 是正向邊
        edges.push_back({ v, 0, cap });
        edges.push_back({ u, 0, 0 });
        adj[u].push_back(m++);
        adj[v].push_back(m++);
    }
    bool bfs() {
        fill(dis.begin(), dis.end(), -1);
        dis[s] = 0; queue<int> q;
        q.push(s);
        while (!q.empty() && dis[t] == -1) {
            int u = q.front(); q.pop();
            for (int id : adj[u]) {
                Edge &e = edges[id];
                if (e.flow == e.cap) continue;
                if (dis[e.to] == -1) {
                    dis[e.to] = dis[u] + 1;
                    q.push(e.to);
                }
            }
        }
        return dis[t] != -1;
    }
    T dfs(int u, T flow) {
        if (flow == 0) return 0;
        if (u == t) return flow;
        for (int
             &cur = ptr[u]; cur < (int)adj[u].size(); cur++) {
            Edge &e = edges[adj[u][cur]];
            if (dis[u] + 1 != dis[e.to]) continue;
            if (e.cap == e.flow) continue;
            T mn = dfs(e.to, min(flow, e.cap - e.flow));
            if (mn > 0) {
                e.flow += mn;
                edges[adj[u][cur] ^ 1].flow -= mn;
                return mn;
            }
        }
        return 0;   // 到不了終點就會 return 0
    }
    T work(int s_, int t_) {
        s = s_; t = t_; T flow = 0;
        while (bfs()) {
            fill(ptr.begin(), ptr.end(), 0);
            while (true) {
                T res = dfs(s, INF_FlOW);
                if (res == 0) break;
                flow += res;
            }
        }
        return flow;
    }
    void reset() {
        for (int i = 0; i < m; i++) edges[i].flow = 0;
    }
};
```

4.2 Min Cut [44ae6c]

```
// CSES Police Chase
int main(){
    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 Hangarian [350fc3]

```
struct Hangarian { // 0-based
    int n, m; // 最小路徑覆蓋，二分匹配
    vector<vector<int>> adj;
    vector<int> used, vis;
    vector<pair<int, int>> match;
    Hangarian(int n_ = 0, int m_ = 0) {
        init(n_, m_);
    }
    void init(int n_ = 0, int m_ = 0) {
        n = n_; m = m_;
        adj.assign(n + m, vector<int>());
        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);
    }
    vector<pair<int, int>> work() {
        match.clear();
        used.assign(n + m, -1);
        vis.assign(n + m, 0);
        auto dfs = [&](auto self, int u) -> bool {
            for (int v : adj[u]) {
                if (vis[v] == 0) {
                    vis[v] = 1;
                    if (used[v] == -1 || self(self, used[v])) {
                        used[v] = u;
                        return true;
                    }
                }
            }
            return false;
        };
        for (int i = 0; i < n; i++) {
            fill(vis.begin(), vis.end(), 0);
            dfs(dfs, i);
        }
        for (int i = n; i < n + m; i++) {
            if (used[i] != -1) {
                match.emplace_back(used[i], i - n);
            }
        }
        return match;
    }
};
```

4.4 MCMF [a7998c]

```
template<class Tf, class Tc>
struct MCMF {
    int n, m, s, t;
    Tf INF_FLOW = numeric_limits<Tf>::max() / 2;
    Tc INF_COST = numeric_limits<Tc>::max() / 2;
    struct Edge {
        int to;
        Tf flow, cap; // 流量跟容量
        Tc cost;
    };
    vector<vector<int>> adj;
    vector<Edge> edges; // 幫每個 edge 編號
    vector<Tc> dis, pot; // johnson algorithm, using spfa
    vector<int> rt; // 路徑恢復，對應 id
    vector<bool> vis;
    MCMF(int n_ = 0) { init(n_); }
    void init(int n_ = 0) {
        n = n_;
        m = 0;
        edges.clear();
        adj.assign(n, vector<int>{});
    }
    void add_edge(int u, int v, Tf cap, Tc cost){
        edges.push_back({v, 0, cap, cost});
        edges.push_back({u, 0, 0, -cost});
        adj[u].push_back(m++);
        adj[v].push_back(m++);
    }
    bool spfa() {
        dis.assign(n, INF_COST);
        rt.assign(n, -1); vis.assign(n, false);
        queue<int> q;
        q.push(s), dis[s] = 0, vis[s] = true;
        while (!q.empty()) {
            int u = q.front(); q.pop();
            vis[u] = false;
            for (int id : adj[u]) {
                auto [v, flow, cap, cost] = edges[id];
                Tc ndis = dis[u] + cost + pot[u] - pot[v];
                if (flow < cap && dis[v] > ndis) {
                    dis[v] = ndis; rt[v] = id;
                    if (!vis[v]) {
                        q.push(v); vis[v] = true;
                    }
                }
            }
        }
        return dis[t] != INF_COST;
    }
    bool dijkstra() {
        dis.assign(n, INF_COST); rt.assign(n, -1);
        priority_queue<pair<Tc, int>,
             vector<pair<Tc, int>>, greater<pair<Tc, int>>> pq;
        dis[s] = 0; pq.emplace(dis[s], s);
        while (!pq.empty()) {
            auto [d, u] = pq.top(); pq.pop();
            if (dis[u] < d) continue;
            for (int id : adj[u]) {
                auto [v, flow, cap, cost] = edges[id];
                Tc ndis = dis[u] + cost + pot[u] - pot[v];
                if (flow < cap && dis[v] > ndis) {
                    dis[v] = ndis; rt[v] = id;
                    pq.emplace(ndis, v);
                }

            }
        }
        return dis[t] != INF_COST;
    }
    // 限定 flow, 最小化 cost
    pair<Tf, Tc> work_flow(int s_, int t_, Tf need) {
        s = s_, t = t_; pot.assign(n, 0);
        Tf flow{}; Tc cost{}; bool fr = true;
        while ((fr ? spfa() : dijkstra())) {
            for (int i = 0; i < n; i++) {
                dis[i] += pot[i] - pot[s];
            }
            Tf f = INF_FLOW;
            for (int i = t; i != s; i = edges[rt[i] ^ 1].to) {
                f = min
                     (f, edges[rt[i]].cap - edges[rt[i]].flow);
            }
            f = min<Tf>(f, need);
            for (int i = t; i != s; i = edges[rt[i] ^ 1].to) {
                edges[rt[i]].flow += f;
                edges[rt[i] ^ 1].flow -= f;
            }
            flow += f; need -= f;
            cost += f * dis[t]; fr = false;
            for (int i = 0; i < n; i++) swap(dis[i], pot[i]);
            if (need == 0) break;
        }
        return make_pair(flow, cost);
    }
    // 限定 cost, 最大化 flow
    pair<Tf, Tc> work_budget(int s_, int t_, Tc budget) {
        s = s_, t = t_; pot.assign(n, 0);
        Tf flow{}; Tc cost{}; bool fr = true;
        while ((fr ? spfa() : dijkstra())) {
            for (int i = 0; i < n; i++) {
                dis[i] += pot[i] - pot[s];
            }
            Tf f = INF_FLOW;
            for (int i = t; i != s; i = edges[rt[i] ^ 1].to) {
                f = min
                     (f, edges[rt[i]].cap - edges[rt[i]].flow);
            }
            f = min<Tf>(f, budget / dis[t]);
            for (int i = t; i != s; i = edges[rt[i] ^ 1].to) {
                edges[rt[i]].flow += f;
                edges[rt[i] ^ 1].flow -= f;
            }
            flow += f; budget -= f * dis[t];
            cost += f * dis[t]; fr = false;
            for (int i = 0; i < n; i++) swap(dis[i], pot[i]);
            if (budget == 0 || f == 0) break;
        }
        return make_pair(flow, cost);
    }
    void reset() {
        for (int i = 0; i < m; i++) edges[i].flow = 0;
    }
};
```

5 String

5.1 KMP [cddfd9]

```
struct KMP {
    string sub;
    vector<int> failure;
    KMP(string sub_) {
        sub = sub_;
        failure.resize(sub.size(), -1);
        buildFailFunction();
    }
    void buildFailFunction() {
        for (int i = 1; i < (int)sub.size(); i++) {
            int now = failure[i - 1];
            while (now != -1
                 && sub[now + 1] != sub[i]) now = failure[now];
            if (sub[now + 1] == sub[i]) failure[i] = now + 1;
        }
    }
    vector<int> match(string &s) {
        vector<int> match;
        for (int i = 0, now = -1; i < (int)s.size(); i++) {
            // now is the compare sucessed length -1
            while (s[i] !=
                 sub[now + 1] && now != -1) now = failure[now];
            // failure stores if comparison fail, move to where
            if (s[i] == sub[now + 1]) now++;
            if (now + 1 == (int)sub.size()) {
                match.push_back(i - now);
                now = failure[now];
            }
        }
        return match;
    }
};
```

5.2 Z Function [8dd6ac]

```
// 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(string s) {
    int n = s.size();
    vector<int> z(n);
    z[0] = 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; // 最後一格不算
}
```

5.3 Duval Algorithm [f9dcca]

```
// 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 min_round(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);
}
```

5.4 Manacher [9c9ca6]

```
// 找到對於每個位置的迴文半徑
vector<int> manacher(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 [3b3aa0]

```
struct Trie {
    struct trie_node {
        bool is_word;
        vector<trie_node *> children;
        trie_node() {
            is_word = false;
            children.resize(26, NULL);
        }
    };
    trie_node *root = new trie_node();
    void insert(string &s) {
        trie_node *cur = root;
        for (int i = 0; i < s.size(); i++) {
            int idx = s[i] - 'a';
            if (cur->children[idx] == NULL) {
                cur->children[idx] = new trie_node();
            }
            cur = cur->children[idx];
        }
        cur->is_word = true;
    }
    bool is_in_trie(string &s) {
        trie_node *cur = root;
        for (int i = 0; i < s.size(); i++) {
            if (cur->
                children[s[i] - 'a'] == nullptr) return false;
            cur = cur->children[s[i] - 'a'];
        }
        return true;
    }
    int search_i_start(string &s, int i, vector<int> &dp) {
        trie_node *cur = root;
        int sz = s.size(), ans = 0;
        for (int j = i; j < sz; j++) {
            if (cur
                ->children[s[j] - 'a'] == nullptr) return ans;
            cur = cur->children[s[j] - 'a'];
            if (cur->is_word)
                (ans += dp[j + 1]) %= mod;
        }
        return ans;
    }
};
int main() {
    // 找到 sub 集合裡，可以重複用，組成 s 的組數
    Trie trie;
    string s; cin >> s;
    int sz = s.size();
    // dp 代表 i 開頭到最後的配對總數
    // 找到有結尾為 stop 的 dp[i] += dp[j + 1]
    int n; cin >> n;
    vector<int> dp(sz + 1, 0);
    for (int i = 0; i < n; i++) {
        string sub; cin >> sub;
        trie.insert(sub);
    }
    dp[sz] = 1;
    for (int i = sz - 1; i >= 0; i--) {
        dp[i] = trie.search_i_start(s, i, dp);
    }
    cout << dp[0] << endl;
}
```

6 Math

6.1 質因數分解 [ee1622]

```
// a^(m-1) = 1 (mod m)
// a^(m-2) = 1/a (mod m)
// EXP2: cout << fast_exp(x, fast_exp(y, p, MOD - 1), MOD)
// Filter + DP; DP save min factor ' recur ' factor decomposition
// FacNums = (x+1)(y+1)(z+1)...
// FacSum = (a^0+a^1...+a^x)(b^0+...+b^y)
// FacMul = N(x+1)(y+1)(z+1)/2

vector<int> is_prime;
// 1 代表是質數，非 1 不是
void init(int n) {
    is_prime.assign(n + 1, 1);
    for (int i = 2; i <= (int)sqrt(n) + 1; i++) {
        if (is_prime[i] == 1) {
            for (int j = i + i; j <= n; j += i) {
                is_prime[j] = i;
            }
        }
    }
}
int main() {
    init(1000000);
    ll ans = 1;
    ll q; cin >> q;
    map<ll, ll> mp;
    while (is_prime[q] != 1) {
        mp[is_prime[q]]++;
        q /= is_prime[q];
    }
    if (q != 1) mp[q]++;
    for (auto [a, b] : mp) {
        ans *= b + 1;
    }
    cout << ans << "\n";
}
```

6.2 模除計算 [9b1014]

```
using i64 = long long;
template<class T>
constexpr T power(T a, i64 b) {
    T res = 1;
    for (; b; b /= 2, a *= a) {
        if (b % 2) {
            res *= a;
        }
    }
    return res;
}

constexpr i64 mul(i64 a, i64 b, i64 p) {
    i64 res = a * b - i64(1.L * a * b / p) * p;
    res %= p;
    if (res < 0) {
        res += p;
    }
    return res;
}
template<i64 P>
struct MLong {
    i64 x;
    constexpr MLong() : x{} {}
    constexpr MLong(i64 x) : x{norm(x % getMod())} {}

    static i64 Mod;
    constexpr static i64 getMod() {
        if (P > 0) {
            return P;
        } else {
            return Mod;
        }
    }
    constexpr static void setMod(i64 Mod_) {
        Mod = Mod_;
    }
    constexpr i64 norm(i64 x) const {
        if (x < 0) {
            x += getMod();
        }
        if (x >= getMod()) {
            x -= getMod();
        }
        return x;
    }
    constexpr i64 val() const {
        return x;
    }
    explicit constexpr operator i64() const {
        return x;
    }
    constexpr MLong operator-() const {
        MLong res;
        res.x = norm(getMod() - x);
        return res;
    }
    constexpr MLong inv() const {
        assert(x != 0);
        return power(*this, getMod() - 2);
    }
    constexpr MLong &operator*=(MLong rhs) & {
        x = mul(x, rhs.x, getMod());
        return *this;
    }
    constexpr MLong &operator+=(MLong rhs) & {
        x = norm(x + rhs.x);
        return *this;
    }
    constexpr MLong &operator-=(MLong rhs) & {
        x = norm(x - rhs.x);
        return *this;
    }
    constexpr MLong &operator/=(MLong rhs) & {
        return *this *= rhs.inv();
    }
    friend constexpr MLong operator*(MLong lhs, MLong rhs) {
        MLong res = lhs;
        res *= rhs;
        return res;
    }
    friend constexpr MLong operator+(MLong lhs, MLong rhs) {
        MLong res = lhs;
        res += rhs;
        return res;
    }
    friend constexpr MLong operator-(MLong lhs, MLong rhs) {
        MLong res = lhs;
        res -= rhs;
        return res;
    }
    friend constexpr MLong operator/(MLong lhs, MLong rhs) {
        MLong res = lhs;
        res /= rhs;
        return res;
    }
    friend
         constexpr istream &operator>>(istream &is, MLong &a) {
        i64 v;
        is >> v;
        a = MLong(v);
        return is;
    }
    friend constexpr
          ostream &operator<<(ostream &os, const MLong &a) {
        return os << a.val();
    }
    friend constexpr bool operator==(MLong lhs, MLong rhs) {
        return lhs.val() == rhs.val();
    }
    friend constexpr bool operator!=(MLong lhs, MLong rhs) {
        return lhs.val() != rhs.val();
    }
};

template<>
i64 MLong<0LL>::Mod = i64(1E18) + 9;

constexpr i64 P = 998244353;
using Z = MLong<P>;
// using Z = MLong<0LL>; // change Mod

struct Comb {
    i64 n;
    vector<Z> _fac;
    vector<Z> _invfac;
    vector<Z> _inv;
    Comb() : n{0}, _fac{1}, _invfac{1}, _inv{0} {}
    Comb(i64 n) : Comb() { init(n); }

    void init(i64 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(i64 m) {
        if (m > n) init(2 * m);
        return _fac[m];
    }
    Z invfac(i64 m) {
        if (m > n) init(2 * m);
        return _invfac[m];
    }
    Z inv(i64 m) {
        if (m > n) init(2 * m);
        return _inv[m];
    }
    Z binom(i64 n, i64 m) {
        if (n < m || m < 0) return 0;
        return fac(n) * invfac(m) * invfac(n - m);
    }
    Z Lucas(Z m, Z n) {
        return n == 0 ? Z(1) : Lucas(m.val() / Z::getMod(),
              n.val() / Z::getMod()) * binom(m.val(), n.val());
    }
} comb; // 注意宣告, 若要換模數需重新宣告
```

6.3 中國餘數定理 [d41d8c]

```
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;
}
// remain, mod
ll CRT(vector<pair<ll, ll>> &a){
    ll prod = 1;
    for (auto x : a) {
        prod *= x.second;
    }
    ll res = 0;
    for (auto x : a) {
        auto t = prod / x.second;
        res += x.first * t % prod * inv(t, x.second) % prod;
        if(res >= prod) res -= prod;
    }
    return res;
}
```

6.4 矩陣與快速冪 [08b5fe]

```cpp
template<class T>
struct Mat {
    int m, n;
    constexpr static ll mod = 1e9 + 7;
    vector<vector<T>> matrix;
    Mat(int n_ = 0) { init(n_, n_); }
    Mat(int m_, int n_) { init(m_, n_); }
    Mat(vector<vector<T>> matrix_) { init(matrix_); }
    void init(int m_, int n_) {
        m = m_; n = n_;
        matrix.assign(m, vector<T>(n));
    }
    void init(vector<vector<T>> &matrix_) {
        m = matrix_.size();
        n = matrix_[0].size();
        matrix = matrix_;
    }
    vector<vector<T>> unit(int n) {   // 單位矩陣
        vector<vector<T>> res(n, vector<T>(n));
        for (int i = 0; i < n; i++) {
            res[i][i] = 1;
        }
        return res;
    }
    constexpr Mat &operator*=(const Mat& rhs) & {
        assert(matrix[0].size() == rhs.matrix.size());
        int m = matrix.size()
            , k = matrix[0].size(), n = rhs.matrix[0].size();
        Mat ans(m, n);
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                for (int l = 0; l < k; l++) {
                    (ans.matrix[i][j] += (matrix[i][
                        l] * rhs.matrix[l][j] % mod)) %= mod;
                }
            }
        }
        matrix = ans.matrix;
        return *this;
    }
    constexpr Mat &operator^=(ll p) & {
        assert(m == n); assert(p >= 0);
        Mat ans(p-- == 0 ? unit(m) : matrix);
        while (p > 0) {
            if (p & 1) ans *= *this;
            *this *= *this;
            p >>= 1;
        }
        matrix = ans.matrix;
        return *this;
    }
    friend Mat operator*(Mat lhs, const Mat &rhs) {
        lhs *= rhs;
        return lhs;
    }
    friend Mat operator^(Mat lhs, const ll p) {
        lhs ^= p;
        return lhs;
    }
};
// fn = fn-3 + fn-2 + fn-1
// 初始矩陣    轉移式
// f4 f3 f2   1 1 0     f5 f4 f3
// f3 f2 f1   1 0 1 =>  f4 f3 f2
// f2 f1 f0   1 0 0     f3 f2 f1
```

6.5 樹論分塊 [06204a]

```cpp
// 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
signed main() {
    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";
}
```

6.6 Mobius Theorem

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

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

    2. μ 是常數函數 1 的反元素
       $\Rightarrow \mu*1=\epsilon$，$\epsilon(n)$只在$n=1$時為 1，其餘情況皆為 0。

  - ϕ 歐拉函數: x 以下與 x 互質的數量

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

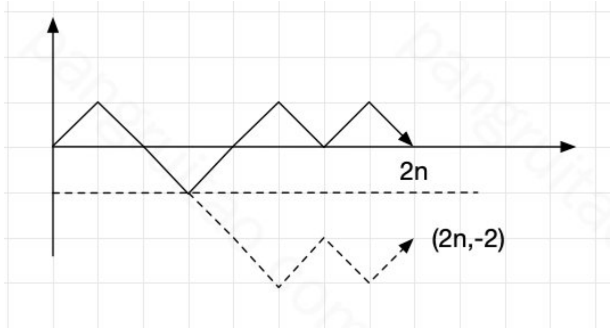
- 莫比烏斯反演公式
  - $f(n)=\sum_{d|n}g(d)\Leftrightarrow g(n)=\sum_{d|n}\mu(d)f(\frac{n}{d})$
  - $f(n)=\sum_{n|d}g(d)\Leftrightarrow g(n)=\sum_{n|d}\mu(\frac{d}{n})f(d)$
- 例子

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

6.7 莫比烏斯反演 [d41d8c]

```cpp
const int maxn = 2e5;
ll mobius_pref[maxn];
void init() {
    mobius_pref[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) {
            mobius_pref[i] = mobius_pref[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]++;
            }
        }
        mobius_pref[i]
            = mobius_pref[i - 1] + (wei[i] % 2 == 0 ? 1 : -1);
    }
}
void solve() {
    ll a, b, c, d, k; cin >> a >> b >> c >> d >> k;
    auto cal = [&](ll x, ll y) -> int {
        int res = 0;
        for (int l = 1, r; l <= min(x, y); l = r + 1) {
            r = min(x / (x / l), y / (y / l));
            res += (mobius_pref[r] - mobius_pref[l
                - 1]) * (x / l) * (y / l);  // 代推出來的式子
        }
        return res;
    };
    cout << cal
        (b / k, d / k) - cal((a - 1) / k, d / k) - cal(b / k,
         (c - 1) / k) + cal((a - 1) / k, (c - 1) / k) << "\n";
}
```

6.8 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

6.9 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 二分搜 [d41d8c]

```
int main() {
    int l = 1, r = 10;
    // 1 to tar, find tar
    while (l <= r) {
        int m = (l + r) / 2;
        if (check(m)) l = m + 1;
        else r = m - 1;
    }
    cout << r;
    // tar to end
    while (l <= r) {
        int m = (l + r) / 2;
        if (check(m)) r = m - 1;
        else l = m + 1;
    }
    cout << l;
}
```

7.2 三分搜 [d41d8c]

```
// 找極值問題，遞增遞減
void solve() {
    int l = 0, r = 10, ans = 0; // ans 紀錄答案
    while (l <= r) {
        int d = (r - l) / 3;     // 差
        int ml = l + d, mr = r - d; // mr 要用減的
        auto cal = [&](int m) -> int {
            int x = 0;
            return x;    // 計算答案
        };
        int ansl = cal(ml), ansr = cal(mr);
        if (ansl < ansr) {
            l = ml + 1;
        }
        else r = mr - 1;
    }
}
```

8 Tree

8.1 LCA [9f95b1]

```
vector<vector<int>> par(maxn, vector<int>(18));
vector<int> depth(maxn + 1);
vector<int> dfn(maxn);
void build_lca(int n, vector<vector<pair<int, int>>> &tree) {
    auto dfs = [&](auto self, int u, int pre) -> void {
        for (auto [v, w] : tree[u]) {
            if (v == pre) continue;
            par[v][0] = u;  // 2 ^ 0
            depth[v] = depth[u] + 1;
            self(self, v, u);
        }
    };
    dfs(dfs, 1, 0);
    for (int i = 1; i <= 18; i++) {
        for (int j = 1; j <= n; j++) {
            par[j][i] = par[par[j][i - 1]][i - 1];
        }
    }
}
int lca(int a, int b) {
    if (depth[a] < depth[b]) swap(a, b);
    int pull = depth[a] - depth[b];
    for (int i = 0; i < 18; i++) {
        if (pull & (1 << i)) {
            a = par[a][i];
        }
    }
    if (a == b) return a;
    for (int i = 17; i >= 0; i--) {
        if (par[a][i] != par[b][i]) {
            a = par[a][i], b = par[b][i];
        }
    }
    return par[a][0];
}
```

8.2 樹重心 [30b436]

```
struct centroid_decomposition {
    int n;
    vector<vector<int>> adj;
    vector<bool> vis;
    vector<int> siz;
    centroid_decomposition() {}
    centroid_decomposition(int n_) { 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 get_siz(int dep, int x, int p = -1) {
        siz[x] = 1;
        for (int y : adj[x]) {
            if (y == p || vis[y]) continue;
            get_siz(dep + 1, y, x);
            siz[x] += siz[y];
        }
    }
    int get_cen(int x, int sz, int p = -1) {
        for (int y : adj[x]) {
            if (y == p || vis[y]) continue;
            if (siz[y] * 2 > sz) {
                return get_cen(y, sz, x);
            }
        }
        return x;
    }
    void work(int x = 0) {
        get_siz(0, x);
        int cen = get_cen(x, siz[x]);
        vis[cen] = true;
        // do something
        for (int y : adj[cen]) {
            if (vis[y]) continue;
            work(y);
        }
    }
};
```

8.3 樹壓平 [51199c]

```
// 父節
     點加值 = 所有子節點區間加值，求單點，使用 bit，做前綴差分
// CSES 1138_Path Queries
int main(){
    int n, q; cin >> n >> q;
    vector<int> node_value(n + 1), euler_ordered_value(n);
    for (int i = 1; i <= n; i++) {
        cin >> node_value[i];
    }
    vector<vector<int>> tree(n + 1);
    for (int i = 1; i < n; i++) {
        int u, v; cin >> u >> v;
        tree[u].push_back(v);
        tree[v].push_back(u);
    }
    vector<pair<int, int>> tree_mapping(n + 1);
    int cnt = 0;
    auto dfs = [&](auto self, int u, int par) -> void {
        euler_ordered_value[++cnt] = node_value[u];
        tree_mapping[u].first = cnt;
        for (auto v : tree[u]) {
            if (v == par) continue;
            self(self, v, u);
        }
        tree_mapping[u].second = cnt;
    };
    dfs(dfs, 1, 0);
    BIT bit(n);
    for (int i = 1; i <= n; i++) {
        bit.modify(tree_mapping[i].first, node_value[i]);
        if (tree_mapping[i].first < n) { // root 就不用扣了
            bit.modify
                (tree_mapping[i].second + 1, -node_value[i]);
        }
    }
    for (int i = 0; i < q; i++) {
        int op; cin >> op;
        if (op == 1) {
            int s, x; cin >> s >> x;
            int add = x
                 - euler_ordered_value[tree_mapping[s].first];
            euler_ordered_value[tree_mapping[s].first] = x;
            bit.modify(tree_mapping[s].first, add);
            if (tree_mapping[s].first < n) { // root 就不用扣了
                bit.modify(tree_mapping[s].second + 1, -add);
            }
        }
        else {
            int node; cin >> node;
            cout <<
                  bit.query(tree_mapping[node].first) << "\n";
        }
    }
}
```

8.4 Heavy Light Decomposition [6791f6]

```
struct HLD {
    int n;
    vector<int> siz, top, dep, parent, in, out, seq;
    vector<vector<int>> adj;
    int cur;

    HLD() {}
    HLD(int n) {
        init(n);
    }
    void init(int n) {
        this->n = n;
        siz.resize(n);
        top.resize(n);
        dep.resize(n);
        parent.resize(n);
        in.resize(n);
        out.resize(n);
        seq.resize(n);
        cur = 0;
        adj.assign(n, {});
    }
    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    void work(int root = 0) {
        top[root] = root;
        dep[root] = 0;
        parent[root] = -1;
        dfs1(root);
        dfs2(root);
    }
    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) {
        // 判斷 u 是否是 v 的祖先
        return in[u] <= in[v] && in[v] < out[u];
    }
    int rootedParent(int u, int v) {
        // 根據新根節點 u 計算 v 的父節點
        swap(u, v);
        if (u == v) {
            return u;
        }
        if (!isAncester(u, v)) {
            return parent[u];
        }
        auto it = upper_bound(adj
             [u].begin(), adj[u].end(), v, [&](int x, int y) {
            return in[x] < in[y];
        }) - 1;
        return *it;
    }
    int rootedSize(int u, int v) {
        // 根據新根節點 u 計算子樹 v 的大小
        if (u == v) {
            return n;
        }
        if (!isAncester(v, u)) {
            return siz[v];
        }
        return n - siz[rootedParent(u, v)];
    }
    int rootedLca(int a, int b, int c) {
        // 根據新的根節點計算三個節點 a、b 和 c 的最近公共祖先
        return lca(a, b) ^ lca(b, c) ^ lca(c, a);
    }
};
```

8.5 Virtual Tree [622e69]

```
// 當存在關鍵點且除了關鍵點的根關鍵點的 LCA 都沒用處
// 可以建立虛樹達成快速樹 DP
// 例如這題是有權樹，跟 vertex 1 隔開的最小成本
int top = -1; vector<int>stk(maxn);
void insert(int u, vector<vector<int>> &vt) {
    if (top == -1) return stk[++top] = u, void();
    int l = lca(stk[top], u);
    if (l == stk[top]) return stk[++top] = u, void();
    while (dfn[l] < dfn[stk[top - 1]])
        vt[stk[top - 1]].push_back(stk[top]), top--;
    if (stk[top - 1] != l) {
        vt[l].push_back(stk[top]);
        stk[top] = l;
    } else vt[l].push_back(stk[top--]);
    stk[++top] = u;
}
void reset(int u, vector<vector<int>> &vt) {
    for (int i : vt[u]) reset(i, vt);
    vt[u].clear();
}
void solve(int n, int q) {
    vector g(n + 1, vector<pair<int, int>>());
    vector vt(n + 1, vector<int>()); // dfs 完清除，否則會退化
    vector<ll> dp(n + 1), iskey(n + 1);
    for (int i = 0; i < n - 1; i++) {
        int u, v, w; cin >> u >> v >> w;
        g[u].push_back({v, w});
        g[v].push_back({u, w});
    }
    build_lca(n, g);
    build(n, g);
    for (int i = 0; i < q; i++) {
        int m; top = -1; cin >> m;
        vector<int> key(m);
        for (int j = 0; j < m; j++) {
            cin >> key[j];
            iskey[key[j]] = 1;
        }
        key.push_back(1);   // 看題目，需要才放
        sort(all(key), [&](int a, int b) {
            return dfn[a] < dfn[b];
        });
        for (int x : key) insert(x, vt);
        while (top
              > 0) vt[stk[top - 1]].push_back(stk[top]), --top;
        // DP
        auto dfs = [&](auto self, int u) -> void {
            for (auto v : vt[u]) {
                self(self, v);
                if (iskey[v]) {
                    dp[u] += min_dis[v];
                    // 砍掉 1 到 v 之間最短的路
                }
                else {
                    dp[u] += min(dp[v], min_dis[v]);
                }
                iskey[v] = dp[v] = 0;
            }
            vt[u].clear();
        };
        dfs(dfs, key[0]); // key[0] 一定是 root
        cout << dp[key[0]] << "\n";
        iskey[key[0]] = dp[key[0]] = 0;
    }
}
```

8.6 Dominator Tree [baa540]

```
struct Dominator_tree {
    int n, id;
    vector<vector<int>> adj, radj, bucket;
    vector<int> sdom, dom, vis, rev, pa, rt, mn, res;
    Dominator_tree(int n_ = 0) { init(n_); }
    void init(int _n) {
        n = _n, id = 0;
        adj.assign(n, vector<int>());
        radj.assign(n, vector<int>());
        bucket.assign(n, vector<int>());
        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]);
        }
    }
    void 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];
    }
};
```

9 DP

9.1 LCS [5781cf]

```
int main() {
    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 [66d09f]

```
int main() {
    int n; cin >> n;
    vector<int> v(n);
    for (int i = 0; i < n; i++) cin >> v[i];
    int dp[n]; vector<int> stk;
    stk.push_back(v[0]);
    dp[0] = 1;  int L = 1;
    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 [308023]

```
int main() {
    string s1, s2; cin >> s1 >> s2;
    int n1 = s1.size(), n2 = s2.size();
    // dp[i][j] 為 s1 的前 i 個字元，跟 s2 的前 j 個字元
    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 [b18541]

```
void travel_exactly_once(){
    // [走過的路][終點]
    vector<vector<int>> dp(1 << 20, vector<int> (20, 0));
    vector<int> rev_adj[20];
    int n, m; cin >> n >> m;
    for(int i = 0; i < m; i++){
        int u, v; cin >> u >> v;
        rev_adj[--v].push_back(--u);
    }
    dp[1][0] = 1;
    for (int road = 0; road < (1 << n); road++) {
        // 沒經過起點，不用走
        if (road & 1 == 0) continue;
        // 有終點但沒全部走過
        if (road & (1
              << (n - 1)) && road != ((1 << n) - 1)) continue;
        // DP，隨便選定一個當前路徑的終點
        for (int end = 0; end < n; end++) {
            // 路徑沒包含假定的 end
            if ((road & (1 << end)) == 0) continue;
            // 去除終點，得到 pre_road
            int pre_road = road - (1 << end);
            // 從 rev_adj 找 pre_road 的終點
            for (int pre_road_end : rev_adj[end]) {
                if ((road & (1 << pre_road_end))) {
                    dp[road
                         ][end] += dp[pre_road][pre_road_end];
                    dp[road][end] %= mod;
                }
            }
        }
    }
    cout << dp[(1 << n) - 1][n - 1];
}
void elevator_rides(){
    int n, k; cin >> n >> k;
    vector<int> passenger(n);
    for (int i = 0; i < n; i++) cin >> passenger[i];
    vector<int
         > used(1 << n, 0);    // 最後載完人的電梯用了多少空間
    vector<int> dp(1 << n, 1);  // bitset
    for (int i = 1; i < 1 << n; i++) {
        used[i] = dp[i] = 2e9;
        for (int j = 0; j < n; j++) {
            if (i & (1 << j)) {  // 有 j
                int pre = i ^ (1 << j);
                // 最後的電梯還能載 j
                if (used[pre] + passenger[j] <= k) {
                    // 電梯數先比，再來比用掉的空間
                    if (dp
                        [pre] < dp[i] || (dp[pre] == dp[i] &&
                        used[pre] + passenger[j] < used[i])) {
                        used[i] = used[pre] + passenger[j];
                        dp[i] = dp[pre];
                    }
                }
                // 搭新的電梯
                else {
                    if (dp[pre] + 1 < dp[i] || (dp[pre] + 1
                        == dp[i] && passenger[j] < used[i])) {
                        used[i] = passenger[j];
                        dp[i] = dp[pre] + 1;
                    }
                }
            }
        }
    }
    cout << dp[(1 << n) - 1];
}
int main(){
    travel_exactly_once();
    elevator_rides();
}
```

9.5 Projects [18998c]

```
// 排程有權重問題，輸出價值最多且時間最少
struct project {
    int from, end, gain, id;
};
int main() {
    int n; cin >> n;
    vector<project> projects(n + 1);
    for (int i = 1; i <= n; i++) {
        int f, e, g; cin >> f >> e >> g;
        projects[i] = {f, e, g, i};
    }
    sort(all(projects), [](project a, project b) {
        if (a.end == b.end) return a.gain < b.gain;
        return a.end < b.end;
    });
    vector<array<int, 3>> dp(n + 1); // nums, gain, time
    vector<int> par(n + 1, 0), ans, add(n + 1, -1);
    for (int i = 1; i <= n; i++) {
        int id = --upper_bound(projects.begin
            (), projects.begin() + i, project({0, projects
            [i].from, 0, 0}), [](project &a, project &b) {
            return a.end < b.end;
        }) - projects.begin();   // 二分搜最接近 from 的 end
        dp[i] = dp[i - 1];
        par[i] = i - 1;
        if (dp
            [i][1] < dp[id][1] + projects[i].gain || (dp[i][1]
             == dp[id][1] + projects[i].gain && dp[i][2] >
            dp[id][2] - projects[i].from + projects[i].end)) {
            // 如果報酬率一樣，比時間少的
            dp[i] = {dp
                [id][0] + 1, dp[id][1] + projects[i].gain, dp
                [id][2] + projects[i].end - projects[i].from};
            par[i] = id;
            add[i] = projects[i].id;
        }
    }
    for (auto i : dp[n])
        cout << i << " " << "\n";
    for (int now = n; now > 0; now = par[now])
        if (add[now] != -1)
            ans.push_back(add[now]);
    sort(all(ans));
    for (auto &i : ans) cout << i << " ";
}
```

9.6 Removal Game [7bb56b]

```
// 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
int main() {
    int n; cin >> n;
    vector<ll> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];
    vector 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 CF Example [7d37ea]

```
// 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
int main() {
    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> l_side(n + 1, inf), cnt(n + 5, 0);
    for (int i = 0; i < m; i++) {
        int l, r; cin >> l >> r;
        l_side[r] = min(l_side[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--) {
        l_side[i - 1] = min(l_side[i - 1], l_side[i]);
    }
    vector<int> dp(n + 1);
    dp[0] = 0;
    for (int i = 1; i <= n; i++) {
        dp[i] = cnt[i];
        if (l_side[i] != inf) {
            dp[i] += dp[l_side[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)
int main(){
    int n, k, ans = 0; cin >> n >> k;
    vector<pii> 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(), [](pii &a, pii &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 << endl;
}
```

10 Geometry

10.1 Basic [d41d8c]

```
#include <bits/stdc++.h>
using namespace std;
using i64 = long long;

template<class T>
struct Point {
    T x;
    T 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>
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>
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(square(p));
}

template<class T>
double length(const Line<T> &l) {
    return length(l.a - l.b);
}

template<class T>
Point<T> normalize(const Point<T> &p) {
    return p / length(p);
}

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>
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>
```

```
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();
    for (int i = 0; i < n; i++) {
        if (pointOnSegment(a, Line(p[i], p[(i + 1) % n]))) {
            return true;
        }
    }

    int t = 0;
    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 : 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>> 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<i64>;
```

10.2 Convex Hull [01a63e]

```
int main() {
    int n; cin >> n;
    vector<P> P(n), U, L;
    for (int i = 0; i < n; i++) {
        cin >> P[i];
    }
    sort(P.begin(), P
         .end(), [](const Point<i64> &a, const Point<i64> &b) {
        return a.x == b.x ? a.y < b.y : a.x < b.x;
    });
    for (int i = 0; i < n; i++) {
        while (L.size() >= 2 && cross(L.back() -
              L[L.size() - 2], P[i] - L[L.size() - 2]) < 0LL) {
            L.pop_back();
        }
        while (U.size() >= 2 && cross(U.back()
             - U[U.size() - 2], P[i] - U[U.size() - 2]) > 0LL){
            U.pop_back();
        }
        L.push_back(P[i]);
        U.push_back(P[i]);
    }
    cout << L.size() + U.size() - 2 << "\n";
    for (int i = 0; i < L.size() - 1; i++) {
        cout << L[i].x << " " << L[i].y << "\n";
    }
    for (int i = U.size() - 1; i > 0; i--) {
        cout << U[i].x << " " << U[i].y << "\n";
    }
}
```

10.3 MinEuclideanDistance [469a8f]

```
template<class T>
T distanceSquare(const Point<T> &a, const Point<T> &b) {
    return square(a - b);
}
void solve() {
    int n; cin >> n;
    constexpr i64 inf = 8e18;
    vector<Point<i64>> a(n);
    for (int i = 0; i < n; i++) {
        i64 x, y;
        cin >> x >> y;
        a[i] = Point<i64>(x, y);
    }
    struct sortY {
        bool operator()
             (const Point<i64> &a, const Point<i64> &b) const {
            return a.y < b.y;
        }
    };
    struct sortXY {
        bool operator()
             (const Point<i64> &a, const Point<i64> &b) const {
            if (a.x == b.x) return a.y < b.y;
            else return a.x < b.x;
        }
    };
    sort(a.begin(), a.end(), sortXY());
    vector<Point<i64>> t(n);
    auto devide = [&](auto &&self, int l, int r) -> i64 {
        if (l == r) return inf;
        int m = (l + r) / 2;
        i64 ans = min(self(self, l, m), self(self, m + 1, r));
        i64 midval = a[m].x;
        i64 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, distanceSquare(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";
}
```

10.4 LatticePoints [7750d6]

```
int main() {
    // Polygun 內整數點數
    int n; cin >> n;
    vector<Point<i64>> polygon(n);
    for (int i = 0; i < n; i++) cin >> polygon[i];
    i64 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<i64>>& polygon) -> i64 {
        i64 res = 0;
        int n = polygon.size();
        for (int i = 0; i < n; i++) {
            i64 dx = polygon[(i + 1) % n].x - polygon[i].x;
            i64 dy = polygon[(i + 1) % n].y - polygon[i].y;
            res += std::gcd(abs(dx), abs(dy));
        }
        return res;
    };
    i64 res = countBoundaryPoints(polygon);
    i64 ans = (area - res + 2) / 2;
    cout << ans << " " << res << "\n";
}
```