

## Contents

<b>1 Basic</b>	<b>1</b>	<b>6 Math</b>	<b>9</b>
1.1 Basic	1	6.1 質因數分解	9
<b>2 Graph</b>	<b>1</b>	6.2 矩陣快速冪	9
2.1 DFS 跟 BFS	1	6.3 盧卡斯定理	9
2.2 DSU	1	6.4 樹論分塊	10
2.3 最短距離算法 - Dijkstra	1	6.5 Theorem	10
2.4 最小生成樹 - Prim	2	<b>7 Search and Greedy</b>	<b>10</b>
2.5 正權找環	2	7.1 二分搜	10
2.6 負權找負環	2	7.2 三分搜	10
2.7 正權最大距離	2	7.3 Concert Ticket	10
2.8 負權最大距離	3	7.4 Restaurant Customers	11
2.9 FloydWarshall	3	<b>8 Tree</b>	<b>11</b>
2.10 歐拉環與歐拉路	3	8.1 LCA	11
2.11 SCC 結合拓模 DP	3	8.2 樹 DFS	11
2.12 2-SAT	4	8.3 樹重心	11
2.13 Planets Cycles	4	8.4 節點距離總和	11
2.14 Planet Queries II	4	8.5 有權樹直徑	11
<b>3 Data Structure</b>	<b>5</b>	8.6 樹壓平	11
3.1 BIT	5	<b>9 DP</b>	<b>12</b>
3.2 Increasing Array Queries	5	9.1 背包問題	12
3.3 線段樹	5	9.2 Bitmask DP	12
3.4 懶標線段樹	6	9.3 硬幣	12
3.5 莫隊	7	9.4 編輯距離	13
3.6 Treap	7	9.5 LCS	13
<b>4 Flow</b>	<b>7</b>	9.6 LIS	13
4.1 Dinic	7	9.7 Projects	13
4.2 MCMF	8	9.8 Removal Game	14
<b>5 String</b>	<b>8</b>	<b>10 Geometry</b>	<b>14</b>
5.1 KMP	8	10.1 Cross Product	14
5.2 Trie	8	10.2 Convex Hull	14

## 1 Basic

### 1.1 Basic [6a5860]

```
// 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

#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#define all(x) (x).begin(), (x).end()
#define pii pair<int, int>
#define endl "\n"
#define int long long
using namespace std;
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>;

struct cmp { // 在有 template 的資料使用
    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 的 upper_bound 跟 lower_bound
    // 功能相反，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，自定義比較結構就比照以上
};

const int llinf = 4e18;
const int inf = 2e9;
const int mod = 1e9 + 7;
const int maxn = 2e5 + 5;

void solve() {
}

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

```
        solve();
    }
}

2 Graph
2.1 DFS 跟 BFS [2a332e]

#include <bits/stdc++.h>
using namespace std;
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> deep(n + 1, 1e9);
    queue<int> q;
    auto bfs = [&](auto self, int u) -> void {
        vis[u] = true;
        deep[u] = 0;
        q.push(u);
        while (!q.empty()) {
            int now = q.front(); q.pop();
            for (auto nxt: adj[now]) {
                if (vis[nxt]) continue;
                vis[nxt] = true;
                deep[nxt] = deep[now] + 1;
                q.push(nxt);
            }
        }
    };
    bfs(bfs, 1);
}
```

### 2.2 DSU [99b9f3]

```
#include <bits/stdc++.h>
using namespace std;
struct DSU {
    vector<int> boss, siz;
    DSU(int n) { // 1 based
        boss.resize(n + 1);
        iota(boss.begin(), boss.end(), 0);
        siz.assign(n + 1, 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;
        return true;
    }
    int size(int x) {
        return siz[find_boss(x)];
    }
};
```

### 2.3 最短距離算法 - Dijkstra [b8dfdb]

```
#include <bits/stdc++.h>
using namespace std;
#define pii pair<int, int>
// Flight Discount
void dijkstra() {
    int n, m; cin >> n >> m;
    vector<vector<pii>> adj(n + 1, vector<pii>(n + 1));
    vector<vector<int>>
        dis(n + 1, vector<int>(2, 2e9)); // 0 for not used
    for (int i = 1; i <= m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        adj[u].push_back({v, w});
    }
    priority_queue
        <array<int, 3>, vector<array<int, 3>>, greater<array
        <int, 3>>> pq; // 0 for w, 1 for u, 2 for discount
    dis[1][0] = dis[1][1] = 0;
    pq.push({0, 1, 0});
    while (!pq.empty()) {
        auto [dist, u, us] = pq.top(); pq.pop();
        if (dis[u][us] < dist) continue;
```

```

    if (us) {
        for (auto [v, w] : adj[u]) {
            if (dis[u][1] + w < dis[v][1]) {
                dis[v][1] = dis[u][1] + w;
                pq.push({dis[v][1], v, 1});
            }
        }
    }
    else {
        for (auto [v, w] : adj[u]) {
            if (dis[u][0] + w < dis[v][0]) {
                dis[v][0] = dis[u][0] + w;
                pq.push({dis[v][0], v, 0});
            }
            if (dis[u][0] + w / 2 < dis[v][1]) {
                dis[v][1] = dis[u][0] + w / 2;
                pq.push({dis[v][1], v, 1});
            }
        }
    }
}
cout << min(dis[n][0], dis[n][1]);
}

```

## 2.4 最小生成樹 - Prim [5318c2]

```

#include <bits/stdc++.h>
using namespace std;
#define pii pair<int, int>
int n, m;
int ans = 0;
const int maxn = 2e5 + 5;
vector<pair<int, int>> adj[maxn];
bool Prim() {
    int node_sz = 0;
    priority_queue<pii, vector<pii>, greater<pii>> pq;
    pq.push({0, 1});
    bool vis[maxn] = {false};
    while (!pq.empty()) {
        auto [cost, u] = pq.top(); pq.pop();
        if (vis[u]) continue;
        vis[u] = true;
        ans += cost;
        node_sz++;
        for (auto [v, cost] : adj[u]) {
            if (!vis[v])
                pq.push({cost, v});
        }
    }
    if (node_sz == n) return true;
    return false;
}
void solve() {
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        int u, v, cost; cin >> u >> v >> cost;
        adj[u].push_back({v, cost});
        adj[v].push_back({u, cost});
    }
    if (Prim()) cout << ans;
    else cout << "IMPOSSIBLE";
}

```

## 2.5 正權找環 [2a579d]

```

#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e5 + 5;
vector<int> graph[maxn];
int color[maxn], parent[maxn];
bool vis[maxn];
int n, m;
void print_ans(int ori) {
    int now = parent[ori];
    deque<int> ans;
    ans.push_front(ori);
    while (now != ori) {
        ans.push_front(now);
        now = parent[now];
    }
    ans.push_front(ori);
    cout << ans.size() << endl;
    for (auto i : ans) {
        cout << i << " ";
    }
    exit(0);
}
void dfs(int now) {
    color[now] = 1;
    vis[now] = 1;
    for (auto nxt : graph[now]) {
        parent[nxt] = now;
        if (color[nxt] == 1) {
            print_ans(nxt);
        }
        else if (color[nxt] == 0) {
            dfs(nxt);
        }
    }
    color[now] = 2;
}
void solve() {
}

```

```

cin >> n >> m;
for (int i = 1; i <= m; i++) {
    int u, v; cin >> u >> v;
    graph[u].push_back(v);
}
for (int i = 1; i <= n; i++) {
    if (!vis[i])
        dfs(i);
}
cout << "IMPOSSIBLE";
}

```

## 2.6 負權找負環 [a27f3b]

```

// 用 Bellman Ford 找負環
#include <bits/stdc++.h>
using namespace std;
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";
    }
}

```

## 2.7 正權最大距離 [a3879a]

```

#include <bits/stdc++.h>
using namespace std;
// 只能用在 DAG，用拓撲按順序鬆弛
// 如果 1 不能到達 n，n 也有
// 可能被鬆弛，所以要注意的是 dis[n] < 0，不能只看有沒有 = -1e9
void print_ans(int n, vector<int> &par) {
    deque<int> ans;
    int now = n;
    while (now != 1) {
        ans.push_front(now);
        now = par[now];
    }
    ans.push_front(1);
    cout << ans.size() << endl;
    for (auto i : ans) {
        cout << i << " ";
    }
}
void solve() {
    int n, m;
    cin >> n >> m;
    vector<int> dis(n + 1, -1e9); dis[1] = 0;
    vector<vector<int>> graph(n + 1, vector<int>());
    vector<bool> vis(n + 1, 0);
    vector<int> par(n + 1);
    vector<int> in(n + 1, 0);
    queue<int> q;
    for (int i = 1; i <= m; i++) {
        int u, v; cin >> u >> v;
        graph[u].push_back(v);
        in[v]++;
    }
    for (int i = 1; i <= n; i++) {
        if (in[i] == 0) q.push(i);
    }
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (auto nxt : graph[u]) {
            if (dis[nxt] < dis[u] + 1) { // 鬆弛
                dis[nxt] = dis[u] + 1;
            }
        }
    }
}

```

```

        par[nxt] = u;
    }
    in[nxt]--;
    if (in[nxt] == 0) q.push(nxt);
}
vis[u] = 1;
}
if (dis[n] < 0) {
    cout << "IMPOSSIBLE";
}
else print_ans(n, par);
}

```

## 2.8 負權最大距離 [8372e8]

```

#include <bits/stdc++.h>
#define int long long
using namespace std;
// CSES High Score
const int maxn = 2505;

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);
    }
}

void bellman_ford(
    int n, int s, vector<int> &vis, vector<int> &dis,
    vector<array<int, 3>> edge, vector<vector<int>> &adj) {
    fill(dis.begin(), dis.end(), -1e18);
    dis[s] = 0;
    for (int i = 1; i <= n; i++) {
        for (auto [u, v, w] : edge) {
            if (dis[u] != -1e18 && dis[v] < dis[u] + w) {
                dis[v] = dis[u] + w;
                if (i == n) {
                    dfs(v, vis, adj);
                }
            }
        }
    }
}

signed main() {
    int n, m; cin >> n >> m;
    vector<array<int, 3>> edge;
    vector<vector<int>> adj(n + 1);
    vector<int> dis(n + 1), vis(n + 1);
    while (m--) {
        int u, v, w;
        cin >> u >> v >> w;
        edge.push_back({u, v, w});
        adj[u].push_back(v);
    }
    bellman_ford(n, 1, vis, dis, edge, adj);
    if (vis[n]) cout << -1;
    else cout << dis[n];
}

```

## 2.9 FloydWarshall [410f48]

```

#include <bits/stdc++.h>
using namespace std;
#define int long long
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";
    }
}

```

## 2.10 歐拉環與歐拉路 [1dc3a5]

```

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

```

```

// 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 << " ";
}

```

## 2.11 SCC 結合拓撲 DP [382a7f]

```

#include <bits/stdc++.h>
using namespace std;
#define int long long
// 找到所有 SCC 然後結合原圖重建一個 DAG，然後拓撲 DP
void dfs(int u, vector<int> &vis, vector<vector<int>> &adj) {
    if (!vis[u]) {
        vis[u] = 1;
        for (auto v : adj[u]) {
            dfs(v, vis, adj);
        }
        adj[u].push_back(u); // finish time 小到大排列
    }
}

void rev_dfs(int u, vector<int> &vis, vector<int> &order, vector<vector<int>> &rev_adj, int &scc_num) {
    if (!vis[u]) {
        vis[u] = 1;
        order[u] = scc_num;
        for (auto v : rev_adj[u]) {
            rev_dfs(v, vis, order, rev_adj, scc_num);
        }
    }
}

signed main() {
    int n, m, scc_num = 0;
    cin >> n >> m;
    vector<int> coin(n + 1), order(n + 1), vis(n + 1, 0);
    vector<vector<int>> adj(n + 1), rev_adj(n + 1);
    vector<int> kosaraju;
    for (int i = 1; i <= n; i++) {
        cin >> coin[i];
    }
    for (int i = 1; i <= m; i++) {
        int u, v; cin >> u >> v;
        adj[u].push_back(v);
        rev_adj[v].push_back(u);
    }
    for (int i = 1; i <= n; i++) {
        if (!vis[i]) {
            dfs(i, vis, kosaraju, adj);
        }
    }
    reverse(kosaraju.begin(), kosaraju.end()); // 轉過來，從 finish time 大的開始做 dfs
    vis.assign(n + 1, 0);
    for (auto &u : kosaraju) {
        if (!vis[u]) {
            scc_num++;
            rev_dfs(u, vis, order, rev_adj, scc_num);
        }
    }
    // 重新建 DAG，根據原圖，如果不再同個 SCC，對 order 加邊
    vector<vector<int>> DAG(scc_num + 1, vector<int>());
    vector<int> in_degree(scc_num + 1, 0);
    vector<int> sum_coin(scc_num + 1, 0), dp_coin(scc_num + 1, 0);
}

```

```

set<pair<int, int>> st;
int ans = -1e9;
for (int i = 1; i <= n; i++) {
    sum_coin[order[i]] += coin[i];
    for (auto j : adj[i]) {
        // 如果不是在同一個 SCC 且 order 邊還沒加過
        if (order[i] != order[j] &&
            st.find({order[i], order[j]}) == st.end()) {
            DAG[order[i]].push_back(order[j]);
            in_degree[order[j]]++;
            st.insert({order[i], order[j]});
        }
    }
}
// 對 DAG 拓撲 DP
queue<int> q;
for (int i = 1; i <= scc_num; i++) {
    if (in_degree[i] == 0) {
        q.push(i);
    }
}
while (!q.empty()) {
    int now = q.front(); q.pop();
    dp_coin[now] += sum_coin[now];
    ans = max(ans, dp_coin[now]);
    for (auto v : DAG[now]) {
        in_degree[v]--;
        dp_coin[v] = max(dp_coin[v], dp_coin[now]);
        if (in_degree[v] == 0) q.push(v);
    }
}
cout << ans;
}

```

## 2.12 2-SAT [7b512c]

```

#include <bits/stdc++.h>
using namespace std;
// +(-) u or +(-) v
const int maxn = 1e5 + 5;
vector<int> adj[2 * maxn], rev_adj[2 * maxn];
vector<int> order;
int cat[2 * maxn];
int k = 1;
bool vis[2 * maxn];
void dfs(int now) {
    if (!vis[now]) {
        vis[now] = 1;
        for (auto v : adj[now]) {
            dfs(v);
        }
        order.push_back(now);
    }
}
void rev_dfs(int now) {
    if (!vis[now]) {
        cat[now] = k;
        vis[now] = 1;
        for (auto v : rev_adj[now]) {
            rev_dfs(v);
        }
    }
}
int main() {
    int n, m;
    cin >> m >> n;
    for (int i = 1; i <= m; i++) {
        int u, v;
        char a, b;
        cin >> a >> u >> b >> v;
        if (a == '-' && b == '-') {
            u = 2 * n - u + 1; // reverse
        }
        if (b == '-' && a == '-') {
            v = 2 * n - v + 1; // reverse
        }
        adj[2 * n - u + 1].push_back(v); // from -u to v; // if -u, then v
        adj[2 * n - v + 1].push_back(u); // from -v to u; // if -v, then u
        rev_adj[v].push_back(2 * n - u + 1);
        rev_adj[u].push_back(2 * n - v + 1);
    }
    for (int i = 1; i <= 2 * n; i++) {
        if (!vis[i]) {
            dfs(i);
        }
    }
    memset(vis, 0, sizeof(vis));
    reverse(order.begin(), order.end());
    for (auto i : order) {
        if (!vis[i]) {
            rev_dfs(i);
            k++;
        }
    }
    char ans[2 * n + 1];
    for (int i = 1; i <= n; i++) {
        if (cat[i] == cat[2 * n - i + 1]) {
            cout << "IMPOSSIBLE";
            return;
        }
    }
}

```

```

    if (cat[i] > cat[2 * n - i + 1]) {
        ans[i] = '+';
    }
    else ans[i] = '-';
}
for (int i = 1; i <= n; i++) {
    cout << ans[i] << " ";
}
}

```

## 2.13 Planets Cycles [391e2a]

```

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

```

## 2.14 Planet Queries II [8c2a64]

```

#include <bits/stdc++.h>
using namespace std;
// now on a and want to reach b, the min steps, directed
int n, q;
const int maxn = 2e5 + 5;
int dp[30][maxn];
vector<vector<int>> cycles;
int no[maxn]; // Order & Can be in cycle, or out
int cycle_idx[maxn];
bool vis[maxn];
void set_out_of_cycle_no(int now, unordered_set<int> &done) {
    if (done.find(now) != done.end()) {
        return;
    }
    set_out_of_cycle_no(dp[0][now], done);
    done.insert(now);
    no[now] = no[dp[0][now]] - 1;
}
int wiint_go_to(int u, int k) { // return the node when walk k
    for (int i = 0; i <= 18; i++) {
        if (k & (1 << i)) {
            u = dp[i][u];
        }
    }
    return u;
}
void find_cycle(int now) {
    unordered_set<int> appear;
    vector<int> vec;
    bool flag = true;
    while (appear.find(now) == appear.end()) {
        appear.insert(now);
        vec.push_back(now);
        if (vis[now]) { // Didn't Find Cycle
            flag = false;
            break;
        }
        now = dp[0][now];
    }
    for (auto i : vec) vis[i] = true;
    if (!flag) return;
    int z = find(vec.begin(), vec.end(), now) - vec.begin(); // start pushing from last now
    int m = vec.size();
    vector<int> cycle;
}

```

```

    for (int i = z; i < m; i++) {
        cycle.push_back(vec[i]);
    }
    cycles.push_back(cycle);
}

void solve() {
    cin >> n >> q;
    for (int u = 1; u <= n; u++) {
        cin >> dp[0][u];
    }
    for (int i = 1; i <= 18; i++) { // Make Chart
        for (int u = 1; u <= n; u++) {
            dp[i][u] = dp[i-1][dp[i-1][u]];
        }
    }
    for (int i = 1; i <= n; i++) {
        if (!vis[i]) find_cycle(i);
    }
    int idx = 0;
    memset(no, -1, sizeof(no));
    memset(cycle_idx, -1, sizeof(cycle_idx));
    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;
        // Same Cycle
        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";
        }
        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;
            }
            int jump = no[v] - no[u];
            if (wiint_go_to(u, jump) == v) {
                cout << jump << "\n";
            }
            else cout << -1 << "\n";
        }
        else if (cycle_idx[u] == -1 && cycle_idx[v] != -1) { // v is in cycle, Smainter Binary Search
            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 != -1 && l <= n) {
                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 is death in the cycle, can't reach
            cout << -1 << "\n";
        }
    }
}

```

## 3 Data Structure

### 3.1 BIT [d41d8c]

```

#include <bits/stdc++.h>
using namespace std;
struct BIT { // BIT 都是 1-based 的查詢
    int n;
    vector<int> bit;
    BIT(int n) { // 有幾個數
        this->n = n;
        bit.resize(n + 1, 0);
    }
    BIT(vector<int> &init) { // 必須是 1-based
        this->n = init.size() - 1;
        bit.resize(n + 1, 0);
        for (int i = 1; i <= n; i++) {
            modify(i, init[i]);
        }
    }
    void modify(int i, int val) {
        for (; i <= n; i += i & -i) {

```

```

            bit[i] += val;
        }
    }
    int query(int r) {
        int ans = 0;
        for (; r; r -= r & -r) ans += bit[r];
        return ans;
    }
    int query(int l, int r) {
        return query(r) - query(l - 1);
    }
};

struct TwoDimensionBIT {
    int nx, ny;
    vector<vector<int>> bit;
    TwoDimensionBIT(int x, int y) {
        nx = x; ny = y;
        bit.resize(x + 1, vector<int>(y + 1, 0));
    }
    void modify(int x, int y, int mod) {
        for (; x <= nx; x += x & -x) {
            for (int tmp = y; tmp <= ny; tmp += tmp & -tmp) {
                bit[x][tmp] += mod;
            }
        }
    }
    int query(int r1, int r2) {
        int ans = 0;
        for (; r1; r1 -= r1 & -r1) {
            for (int tmp = r2; tmp <= ny; tmp -= tmp & -tmp) {
                ans += bit[r1][tmp];
            }
        }
        return ans;
    }
};

```

### 3.2 Increasing Array Queries [d41d8c]

```

#include <bits/stdc++.h>
using namespace std;
const int maxn = 2e5 + 5;
int n, q;
int nums[maxn], prefix[maxn], ans[maxn], BIT[maxn], contrib[maxn];
vector<pair<int, int>> queries[maxn];
void update(int pos, int val) {
    for (; pos <= n; pos += pos & -pos) BIT[pos] += val;
}
int query(int a, int b) {
    int ans = 0;
    for (; b; b -= b & -b) ans += BIT[b];
    for (a--; a; a -= a & -a) ans -= BIT[a];
    return ans;
}
void solve() {
    cin >> n >> q;
    for (int i = 1; i <= n; i++) {
        cin >> nums[i];
        prefix[i] = prefix[i-1] + nums[i];
    }
    nums[n + 1] = 1e9;
    prefix[n + 1] = 2e18;
    for (int i = 1; i <= q; i++) {
        int a, b; cin >> a >> b;
        queries[a].push_back({b, i});
    }
    deque<int> mono; mono.push_front(n + 1);
    for (int i = n; i > 0; i--) { // question from start at n to start at 1
        while (nums[i] >= nums[mono.front()]) {
            update(mono.front(), -contrib[mono.front()]); // mono.front's contrib become 0
            mono.pop_front();
        }
        contrib[i] = (mono.front() - 1 - i) *
            nums[i] - (prefix[mono.front() - 1] - prefix[i]);
        update(i, contrib[i]);
        mono.push_front(i);
        for (auto j : queries[i]) { // pos is the index in mono <= end's
            int pos = upper_bound(mono.begin(), mono.end(), j.first) - mono.begin() - 1;
            ans[j.second] = (pos ? query(i, mono[pos - 1]) : 0) // smainter than y's mono
                // mono to y caculate directly
                + (j.first - mono[pos]) * nums[mono[pos]] - (prefix[j.first] - prefix[mono[pos]]);
        }
    }
    for (int i = 1; i <= q; i++) {
        cout << ans[i] << endl;
    }
}

```

### 3.3 線段樹 [d41d8c]

```

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

```

```

struct Seg {
    int n;
    vector<Node> tree;
    Seg (vector<Node> init_) {
        n = init_.size() - 1;
        tree.resize(4 * n);
        function <void(int
            , int, int)> build = [&](int now, int l, int r) {
            if (l == r) {
                tree[now] = init_[l];
                return;
            }
            int m = (l + r) / 2;
            build(now << 1, l, m);
            build((now << 1) + 1, m + 1, r);
            pull(now);
        };
        build(1, 1, n);
    }
    Node query(int l, int r, int ql, int qr, int now) {
        int m = (l + r) >> 1;
        if (qr < l || ql > r) {
            return Node();
        }
        if (ql <= l && r <= qr) {
            return tree[now];
        }
        return query(l, m, ql, qr, now
            << 1) + query(m + 1, r, ql, qr, (now << 1) + 1);
    }
    Node query(int l, int r) { return query(1, n, l, r, 1); }
    void pull(int now) {
        tree[now] = tree[now << 1] + tree[(now << 1) + 1];
    }
    void modify(int l, int r, int idx, int now, int add) {
        if (l == r) {
            tree[now].sum = add;
        }
        // -----
        // how to modify ?-----
        // -----
        return;
    }
    int m = (l + r) >> 1;
    if (idx <= m) {
        modify(l, m, idx, now << 1, add);
    }
    else {
        modify(m + 1, r, idx, (now << 1) + 1, add);
    }
    pull(now);
}
void modify
    (int idx, int add) { modify(1, n, idx, 1, add); }
};
// -----
// define structure and info plus-----
struct Node {
    int sum;
    Node () {
        sum = 0;
    }
};
Node operator + (const Node &a, const Node &b) {
    Node c;
    c.sum = a.sum + b.sum;
    return c;
    // use lc\rc to undate now
    // tree[now].sum = tree[lc].sum + tree[rc].sum;
    // tree[now].prefix
    // = max(tree[lc].sum+tree[rc].prefix, tree[lc].prefix);
    // tree[now].suffix
    // = max(tree[lc].suffix+tree[rc].sum, tree[rc].suffix);
    // tree[now].middle_max = max(max(tree[lc].middle_max, tree
    // [rc].middle_max), tree[lc].suffix+tree[rc].prefix);
    // tree[now].middle_max = max(max(tree[
    // now].middle_max, tree[now].prefix), tree[now].suffix);
}
// -----
// -----
// pizza_queries
// 左邊的店(s < t): dis_l = (pizza[s] - s) + t;
// 右邊的店(t < s): dis_r = (pizza[s] + s) - t;
// 實作：建左查詢線段樹跟右查詢線段樹，用最小值pull
// 答案是 min(left_query(1, s) + t, right_query(s, end) + t);
// List Removals
// 維護區間內有幾個數字被選過
// 用二
// 分搜找右區間最小位，使得 ans - query == 1~ans 被選過的數量

```

### 3.4 懶標線段樹 [d41d8c]

```

#include <bits/stdc++.h>
using namespace std;
template <class Node, class Lazy>

```

```

struct LazySeg {
    int n;
    vector<Node> tree;
    vector<Lazy> lazy;
    template <typename T>
    LazySeg (vector<T> init_) { // 必須是 1-based
        n = init_.size() - 1;
        tree.resize(4 * n);
        lazy.resize(4 * n);
        function <void(int
            , int, int)> build = [&](int now, int l, int r) {
            if (l == r) {
                tree[now] = init_[l];
                return;
            }
            int m = (l + r) / 2;
            build(now << 1, l, m);
            build((now << 1) + 1, m + 1, r);
            pull(now);
        };
        build(1, 1, n);
    }
    Node query(int l, int r, int ql, int qr, int now) {
        int m = (l + r) >> 1;
        if (qr < l || ql > r) {
            // -----
            // out of range, return what-----
            // -----
            return Node();
        }
        }
        push(now, l, r);
        if (ql <= l && r <= qr) {
            return tree[now];
        }
        return query(l, m, ql, qr, now
            << 1) + query(m + 1, r, ql, qr, (now << 1) + 1);
    }
    Node query(int l, int r) { return query(1, n, l, r, 1); }
    void pull(int now) {
        tree[now] = tree[now << 1] + tree[(now << 1) + 1];
    }
    void modify_add
        (int l, int r, int ql, int qr, int now, int add) {
        int m = (l + r) >> 1;
        if (qr < l || ql > r) {
            return;
        }
        if (ql <= l && r <= qr) {
            // -----
            // how to modify ?-----
            // -----
            lazy[now].add += add;
        }
        return;
    }
    push(now, l, r);
    modify_add(l, m, ql, qr, now << 1, add);
    modify_add(m + 1, r, ql, qr, (now << 1) + 1, add);
    push(now << 1, l, m);
    push((now << 1) + 1, m + 1, r);
    pull(now);
}
void modify_add(int
    l, int r, int add) { modify_add(1, n, l, r, 1, add); }
void modify_set
    (int l, int r, int ql, int qr, int now, int val) {
    int m = (l + r) >> 1;
    if (qr < l || ql > r) {
        return;
    }
    if (ql <= l && r <= qr) {
        // -----
        // how to modify ?-----
        // -----
        lazy[now].set_val = val;
        lazy[now].add = 0;
    }
    return;
    }
    push(now, l, r);
    modify_set(l, m, ql, qr, now << 1, val);
    modify_set(m + 1, r, ql, qr, (now << 1) + 1, val);
    push(now << 1, l, m);
    push((now << 1) + 1, m + 1, r);
    pull(now);
}
void modify_set(int
    l, int r, int val) { modify_set(1, n, l, r, 1, val); }
void push(int now, int l, int r) {
    apply(now, l, r);
}
// -----
// how to push down ?-----
// -----
if (l != r) {
    if (lazy[now].set_val) {
        lazy[now << 1].set_val = lazy[now].set_val;
    }
}

```



```

        lazy[(now
            << 1) + 1].set_val = lazy[now].set_val;
        lazy[now << 1].add = lazy[now].add;
        lazy[(now << 1) + 1].add = lazy[now].add;
    }
    else {
        lazy[now << 1].add += lazy[now].add;
        lazy[(now << 1) + 1].add += lazy[now].add;
    }
}

// -----

lazy[now] = Lazy();
}
void apply(int now, int l, int r) {
    if (lazy[now].set_val) {
        tree[now].sum = (r - l + 1) * lazy[now].set_val;
    }
    tree[now].sum += (r - l + 1) * lazy[now].add;
}
};
// -----
// define structure and info plus-----
struct Node {
    int sum;
};
struct Lazy {
    int set_val; int add;
};
Node operator+(const Node &a, const Node &b) {
    return {{a.sum + b.sum}};
}
// -----

// polynomial queries
// 設置梯形的底跟加了幾次, apply_tag時底為
// l的合, d為加給次, 所以sum += (底*2 + 次*區間) * 區間 / 2;

```

### 3.5 莫隊 [d41d8c]

```

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

```

### 3.6 Treap [d41d8c]

```

#include <bits/stdc++.h>
using namespace std;
struct Treap {
    Treap *l, *r;
    int pri, subsize; char val; bool rev_valid;
    Treap(int val) {
        this->val = val;
        pri = rand();
        l = r = nullptr;
        subsize = 1; rev_valid = 0;
    }
    void pull() { // update subsize or other information
        subsize = 1;
        for(auto i : {l, r}) {
            if (i) subsize += i->subsize;
        }
    }
};
int size(Treap *treap) {
    if (treap == NULL) return 0;
    return treap->subsize;
}
// lazy
void push(Treap *t) {
    if (!t) return;
    if (t->rev_valid) {
        swap(t->l, t->r);
        if (t->l) t->l->rev_valid ^= 1;
        if (t->r) t->r->rev_valid ^= 1;
    }
}

```

```

    }
    t->rev_valid = false;
}
Treap *merge(Treap *a, Treap *b) {
    if (!a || !b) return a ? a : b;
    // push(a); push(b); // lazy
    if (a->pri > b->pri) {
        a->r = merge(a->r, b); // a->r = new, inorder, make sense
        a->pull();
        return a;
    }
    else {
        b->l = merge(a, b->l); // new->l = a, inorder, make sense
        b->pull();
        return b;
    }
}
pair<Treap*, Treap*> split(Treap *root, int k) { // find 1~k
    if (root == nullptr) return {nullptr, nullptr};
    // push(root); // lazy
    if (size(root->l) < k) {
        auto [a, b] = split(root->r, k - size(root->l) - 1);
        root->r = a;
        root->pull();
        return {root, b};
    }
    else {
        auto [a, b] = split(root->l, k);
        root->l = b;
        root->pull();
        return {a, root};
    }
}
void Print(Treap *t) {
    if (t) {
        // push(t); // lazy
        Print(t->l);
        cout << t->val;
        Print(t->r);
    }
}
void substring_rev() {
    int n, m; cin >> n >> m;
    Treap *root = nullptr;
    string str; cin >> str;
    for(auto c : str) {
        root = merge(root, new Treap(c));
    }
    for(int i = 1; i <= m; i++) {
        int x, y; cin >> x >> y;
        auto [a, b] = split(root, x-1); // a: 1~x-1, b: x~n
        auto [c, d] = split(b, y-x+1); // Use b to split
        // c->rev_valid ^= true;
        // push(c);
        b = merge(a, d); // Notice the order
        root = merge(b, c);
    }
    Print(root);
}

```

## 4 Flow

### 4.1 Dinic [4d1a72]

```

#include <bits/stdc++.h>
using namespace std;
bool vis[505];
int lev[505], n, m, ans;
struct edge {
    int to, w, rev_ind;
};
vector<edge> adj[505];
bool label_level() {
    // Tag the depth, if can't reach end => return false
    memset(lev, -1, sizeof(lev));
    lev[1] = 0;
    queue<int> q; q.push(1);
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (auto i : adj[u]) {
            if (i.w > 0 && lev[i.to] == -1) {
                q.push(i.to);
                lev[i.to] = lev[u] + 1;
            }
        }
    }
    return (lev[n] == -1 ? false : true);
}
int dfs(int u, int flow) {
    if (u == n) return flow;
    for (auto &i : adj[u]) {
        if (lev[i.to] == lev[u] + 1 && !vis[i.to] && i.w > 0) {
            vis[i.to] = true;
            int ret = dfs(i.to, min(flow, i.w));
            if (ret > 0) {
                i.w -= ret;
                adj[i.to][i.rev_ind].w += ret;
                return ret;
            }
        }
    }
}

```

```

    }
}
return 0; // if can't reach end => return 0
}
void dinic() {
    while (label_level()) {
        while (1) {
            init(v, 0);
            int tmp = dfs(1, inf);
            if (tmp == 0) break;
            ans += tmp;
        }
    }
}
void build() {
    for (int i = 1; i <= m; i++) {
        int u, v, w; cin >> u >> v >> w;
        adj[u].push_back({v, w, (int)adj[v].sz}); // inverse flow's index
        adj[v].push_back({u, 0, (int)adj[u].sz - 1}); // have pushed one, need to -1
    }
}
// Police Chase, need
// to open adj to Augment && ori to determine what pb give
// Dinic \ dfs2, then use reach as u, if the edge
// pb has given && w == 0 && v is not in reach, is the ans
void dfs2(int now, unordered_set<int> &reach) {
    if (!vis[now]) {
        vis[now] = 1;
        reach.insert(now);
        for (auto i : adj[now]) {
            if (i.w > 0) {
                dfs2(i.to, reach);
            }
        }
    }
}
// two two pair // School Dance
// Dinic, then w == 0's edge, which pb has given is the ans

// Distinct Route
// edge set valid var, if we need
// to argument pos road, the reverse edge set true valid;
// if we need argument the argumented
// edge, both set false. Last, from v dfs ans times
bool get_road(int now, vector<int> &ans, vector<bool> &vis) {
    if (now == 1) return true;
    for (auto &v : adj[now]) {
        if (v.arg_valid && !vis[v.to]) {
            ans.push_back(v.to);
            vis[v.to] = true;
            bool flag = get_road(v.to, ans, vis);
            if (flag) {
                v.arg_valid = false;
                return true;
            }
            ans.pop_back();
        }
    }
    return false;
}
}
}

```

## 4.2 MCMF [40d5b7]

```

// Ceiled MinCostMaxFlow, if not, use dinic
typedef struct {
    int from, to, w, cost;
} edge;
int n, m, parcel;
vector<edge> adj; // set num to each edge
vector<int> p[505]; // p[u] has edge's num
int now_edge = 0;
void add_edge(int u, int v, int w, int cost) {
    adj.push_back({u, v, w, cost});
    p[u].push_back(now_edge);
    now_edge++;
    adj.push_back({v, u, 0, -cost}); // argumenting path use -
    p[v].push_back(now_edge);
    now_edge++;
}
ll Bellman_Ford() {
    vector<ll> dis(n+1, inf); dis[1] = 0;
    vector<int> par(m);
    vector<int> flow_rec(n+1, 0); flow_rec[1] = 1e9;
    for (int i = 1; i < n; i++) {
        bool flag = 1;
        int size = adj.sz;
        for (int i = 0; i < size; i++) {
            auto &[from, to, w, cost] = adj[i];
            if (w > 0 && dis[to] > dis[from] + cost) {
                flag = 0;
                dis[to] = dis[from] + cost;
                par[to] = i; // record num
                flow_rec[to] = min(flow_rec[from], w);
            }
        }
        if (flag) break;
    }
    if (dis[n] == 1e9) return 0;
    int mn_flow = flow_rec[n];
}

```

```

int v = n;
while (v != 1) {
    int u = adj[par[v]].from;
    adj[par[v]].w -= mn_flow;
    adj[par[v] ^ 1].w += mn_flow;
    v = u;
}
mn_flow = min(mn_flow, parcel);
parcel -= mn_flow;
return mn_flow * dis[n];
}
void solve() {
    cin >> n >> m >> parcel;
    ll ans = 0;
    for (int i = 1; i <= m; i++) {
        int u, v, w, cost; cin >> u >> v >> w >> cost;
        add_edge(u, v, w, cost);
    }
    while (parcel > 0) {
        int tmp = Bellman_Ford();
        if (tmp == 0) break;
        ans += tmp;
    }
    cout << (parcel > 0 ? -1 : ans);
}

```

## 5 String

### 5.1 KMP [11be97]

```

#include <bits/stdc++.h>
using namespace std;
struct KMP {
    string sub;
    vector<int> failure;
    KMP(string &sub) {
        this->sub = sub;
        failure.resize(sub.size(), -1);
        buildFailFunction();
    }
    void buildFailFunction() {
        for (int i = 1; i < 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> KMPmatching(string &s) {
        vector<int> match;
        for (int i = 0, now = -1; i < s.size(); i++) {
            // now is the compare suceeded length -1
            while (s[i] != sub[now + 1] && now != -1) now = failure[now];
            // f stores if comparison fail, move to where
            if (s[i] == sub[now + 1]) now++;
            if (now + 1 == sub.size()) {
                match.push_back(i - now);
                now = failure[now];
            }
        }
        return match;
    }
};
int main() {
    string s = "xtxtxtxtx";
    string sub = "tx";
    KMP kmp(sub);
    vector<int> ans = kmp.KMPmatching(s);
    for (auto &i : ans) cout << i << " ";
}

```

### 5.2 Trie [b84198]

```

#include <bits/stdc++.h>
using namespace std;
#define all(x) (x).begin(), (x).end()
#define endl "\n"
#define int long long
typedef pair<int, int> pii;
const int llinf = 4e18;
const int inf = 2e9;
const int mod = 1e9 + 7;
const int maxn = 2e5 + 5;

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

```

## 6 Math

### 6.1 質因數分解 [b535c8]

```

#include <bits/stdc++.h>
using namespace std;
// a^(m-1) {\triple_equal} 1 (mod m)
// a^(m-2) {\triple_equal} 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
int main() {
    vector<int> is_prime(2e6 + 1, 1);
    // 1 代表是質數，非 1 不是
    for (int i = 2; i <= 1000; i++) {
        if (is_prime[i] == 1) {
            for (int j = i + i; j <= 1000000; j += i) {
                is_prime[j] = 0;
            }
        }
    }
    int ans = 1;
    int q; cin >> q;
    map<int, int> 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 矩陣快速冪 [d41d8c]

```

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

const int mod = 1e9 + 7;

struct Mat {

```

```

    int n;
    vector<vector<int>> matrix;
    Mat(int n) {
        this->n = n;
        matrix.resize(n);
        for (int i = 0; i < n; i++) {
            matrix[i].resize(n);
        }
    }
    Mat(vector<vector<int>> matrix) {
        this->n = matrix.size();
        this->matrix = matrix;
    }
    Mat unit(int n) { // 單位矩陣
        Mat res(n);
        for (int i = 0; i < n; i++) {
            res.matrix[i][i] = 1;
        }
        return res;
    }
    void mul(Mat b) {
        Mat ans(n);
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                for (int k = 0; k < n; k++) {
                    (ans.matrix[i][j] += (matrix[i][k] * b.matrix[k][j] % mod)) %= mod;
                }
            }
        }
        matrix = ans.matrix;
    }
    void pow(int p) {
        Mat x = *this;
        *this = unit(n);
        while (p > 0) {
            if (p & 1) {
                mul(x);
            }
            x.mul(x);
            p >>= 1;
        }
    }
};
signed main() {
    int n, ans; cin >> n;
    if (n <= 4) {
        vector<int> v = {0, 1, 1, 2, 4};
        ans = v[n];
    }
    else {
        Mat mat({{4, 2, 1}, {2, 1, 1}, {1, 1, 0}});
        Mat x(3);
        x.matrix = {{1, 1, 0}, {1, 0, 1}, {1, 0, 0}};
        x.pow(n - 4);
        mat.mul(x);
        ans = mat.matrix[0][0];
    }
    cout << ans << "\n";
}
// 初始矩陣
// f4 f3 f2
// f3 f2 f1
// f2 f1 f0

// 轉移式
// 1 1 0
// 1 0 1
// 1 0 0

// ==>
// f5 f4 f3
// f4 f3 f2
// f3 f2 f1

```

### 6.3 盧卡斯定理 [cf624d]

```

#include <bits/stdc++.h>
using namespace std;
struct nCr {
    int mod;
    nCr(int mod) : mod(mod) {};
    int inverse(int num) {
        if (num == 1) return 1;
        return (mod - ((mod / num) * inverse(mod % num)) % mod) % mod;
    }
    int fast_exp(int x, int p) {
        int ans = 1;
        while (p > 0) {
            if (p & 1) ans = (ans * x) % mod;
            x = x * x % mod;
            p >>= 1;
        }
        return ans;
    }
};
vector<int> fac;
void BuildLucas(int n) {
    fac.resize(n + 1);
    fac[0] = 1;
    for (int i = 1; i <= n; i++) {
        fac[i] = fac[i - 1] * i % mod;
    }
}

```

```

    }
}
int C(int m, int n) {
    return m < n ? 0 : fac[m] *
        inverse(fac[n]) % mod * inverse(fac[m - n]) % mod;
}
int Lucas(int m, int n) {
    return n == 0 ? 1 % mod : Lucas
        (m / mod, n / mod) * C(m % mod, n % mod) % mod;
}
};

```

## 6.4 樹論分塊 [fe6b55]

```

// CSES_Sum_of_Divisors
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int mod = 1e9 + 7;
const int inv_2 = 500000004;
// n / 1 * 1 + n / 2 * 2 + n / 3 * 3 + ... + n / n * n
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int ans = 0;
    int n; cin >> n;
    for (int l = 1, r; l <= n; l = r + 1) {
        r = n / (n / l);
        int val = n / l; // n / l 到 n / r 一樣的值
        int 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";
}

```

## 6.5 Theorem

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

- 莫比烏斯函數

1. 定義

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

2.  $\mu$  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。

-  $\phi$  歐拉函數:  $x$  以下與  $x$  互質的數量

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

• 莫比烏斯反演公式

$$\begin{aligned}
 - f(n) &= \sum_{d|n} g(d) \Leftrightarrow g(n) = \sum_{d|n} \mu(d) f\left(\frac{n}{d}\right) \\
 - f(n) &= \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu\left(\frac{d}{n}\right) f(d)
 \end{aligned}$$

• 例子

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

## 7 Search and Greedy

### 7.1 二分搜 [d41d8c]

```

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

```

### 7.3 Concert Ticket [d41d8c]

```

// Better than Binary Search
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n, m; cin >> n >> m;
    multiset<int> tik;
    for (int i = 0; i < n; i++) {
        int tmp; cin >> tmp;
        tik.insert(tmp);
    }
    while (m--) {
        int x; cin >> x;
        auto it = tik.upper_bound(x);
        if (it == tik.begin()) {
            cout << -1 << " ";
            continue;
        }
        it--;
        cout << *it << " ";
        tik.erase(it);
    }
}

```

## 7.4 Restaurant Customers [d41d8c]

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    vector<pair<int, int>> times;
    int n; cin >> n;
    for (int i = 0; i < n; i++) {
        int u, v; cin >> u >> v;
        times.push_back({u, 1});
        times.push_back({v, -1});
    }
    sort(times.begin(), times.end());
    int now_people = 0, ans = 0;
    for (auto [t, x] : times) {
        ans = max(ans, (now_people += x));
    }
    cout << ans;
}
```

## 8 Tree

### 8.1 LCA [ca194c]

```
#include <bits/stdc++.h> // LCA from 1
using namespace std;
int main() {
    int n, q; cin >> n >> q;
    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<vector<int>> par(n + 1, vector<int>(18));
    vector<int> depth(n + 1);
    auto dfs = [&](auto self, int u, int pre) -> void {
        for (auto v : 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];
        }
    }
    auto lca = [&](int a, int b) -> int {
        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 DFS [b6cb9a]

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 2e5+5;
vector<int> depth;
void dfs(vector<vector<int>> &tree, int u, int pre) {
    for (auto v : tree[u]) {
        if (v == pre) continue;
        depth[v] = depth[u] + 1;
        dfs(tree, v, u);
    }
}
```

### 8.3 樹重心 [2771f3]

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 2e5+5;
vector<int> tree[maxn];
int cen = 0, n;
int dfs(int par, int now) {
    bool flag = 1;
    int size = 0;
    for (auto nxt : tree[now]) {
        if (par != nxt) {
            int subsize = dfs(now, nxt);
            if (subsize > n / 2) flag = false;
            size += subsize;
        }
    }
    if (n - 1 - size > n / 2) flag = false;
    if (flag) cen = now;
}
```

```
return size + 1;
}
int main() {
    cin >> n;
    for (int i = 1; i < n; i++) {
        int u, v; cin >> u >> v;
        tree[u].push_back(v);
        tree[v].push_back(u);
    }
    for (int i = 1; i <= n; i++) {
        for (auto nxt : tree[i])
            dfs(i, nxt);
        if (cen) break;
    }
}
```

### 8.4 節點距離總和 [3bfb86]

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 2e5+5;
vector<int> tree[maxn];
vector<int> subtree(maxn, 1);
long long ans[maxn];
int n;
void dfs(int par, int now, int depth) {
    ans[1] += depth;
    for (auto nxt : tree[now]) {
        if (par != nxt) {
            dfs(now, nxt, depth + 1);
            subtree[now] += subtree[nxt];
        }
    }
}
void find_ans(int par, int now) {
    // each sub's dis make - 1, non subnode + 1
    for (auto nxt : tree[now]) {
        if (par != nxt) {
            ans[nxt] =
                ans[now] + (n - subtree[nxt]) - subtree[nxt];
            find_ans(now, nxt);
        }
    }
}
int main() {
    cin >> n;
    for (int i = 1; i < n; i++) {
        int u, v; cin >> u >> v;
        tree[u].push_back(v);
        tree[v].push_back(u);
    }
    dfs(0, 1, 0);
    find_ans(0, 1);
    for (int i = 1; i <= n; i++) {
        cout << ans[i] << " ";
    }
}
```

### 8.5 有權樹直徑 [98f093]

```
#include <bits/stdc++.h> // weighted tree centroid
using namespace std;
const int maxn = 1e5+5;
using ll = long long;
vector<pair<int, int>> tree[maxn];
ll dp[maxn];
ll ans = 0;
void DP(int now, int par) {
    ll mx1 = 0; ll mx2 = 0;
    for (auto [nxt, w] : tree[now]) {
        if (nxt == par) continue;
        DP(nxt, now);
        if (mx1 < w + dp[nxt]) { // mx2 = mx1, mx1 = new mx
            mx2 = mx1; mx1 = w + dp[nxt];
        }
        else if (mx2 < w + dp[nxt]) { // mx2 = new
            mx2 = w + dp[nxt];
        }
    }
    dp[now] = mx1;
    ans = max(ans, mx1 + mx2);
}
int main() {
    int n; cin >> n;
    memset(dp, 0, sizeof(dp));
    for (int i = 1; i < n; i++) {
        int u, v, w; cin >> u >> v >> w;
        tree[u].push_back({v, w});
        tree[v].push_back({u, w});
    }
    DP(1, 0);
    cout << (ans < 0 ? 0 : ans);
}
```

### 8.6 樹壓平 [03946b]

```
// 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
#include <bits/stdc++.h>
#define int long long
```

```

using namespace std;
struct BIT { // BIT 都是 1-based 的查詢
    int n;
    vector<int> bit;
    BIT(int n) { // 有幾個數
        this->n = n;
        bit.resize(n + 1, 0);
    }
    BIT(vector<int> &init) { // 必須是 1-based
        this->n = init.size() - 1;
        bit.resize(n + 1, 0);
        for (int i = 1; i <= n; i++) {
            modify(i, init[i]);
        }
    }
    void modify(int i, int val) {
        for (; i <= n; i += i & -i) {
            bit[i] += val;
        }
    }
    int query(int r) {
        int ans = 0;
        for (; r; r -= r & -r) ans += bit[r];
        return ans;
    }
    int query(int l, int r) {
        return query(r) - query(l - 1);
    }
};

void solve() {
    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) {
            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) {
                bit.modify(tree_mapping[s].second + 1, -add);
            }
        }
        else {
            int node; cin >> node;
            cout <<
                bit.query(tree_mapping[node].first) << "\n";
        }
    }
}

```

## 9 DP

### 9.1 背包問題 [9457ef]

```

#include <bits/stdc++.h>
using namespace std;
// 考慮前 i 個，預算有 j 塊錢的最多 page
int main() {
    int n, bud;
    cin >> n >> bud;
    vector<vector<int>> dp(n + 1, vector<int>(bud + 1));
    vector<int> Page(n + 1, 0);
    vector<int> Price(n + 1, 0);

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

```

```

for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= bud; j++) {
        if (j >= Price[i]) { // 買得起
            // 不買或買
            dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - Price[i]] + Page[i]);
        }
        else {
            dp[i][j] = dp[i - 1][j];
        }
    }
}
cout << dp[n][bud] << "\n";
}

```

### 9.2 Bitmask DP [c130ec]

```

#include <bits/stdc++.h>
using namespace std;
const int mod = 1e9 + 7;
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();
}

```

### 9.3 硬幣 [d41d8c]

```

#include <bits/stdc++.h>
using namespace std;
const int mod = 1e9 + 7;

```

```

void coin_combination_II(){
    // 有 n 種錢幣，求組合為 x 的組數，順序不可顛倒
    // 可顛倒的話只要一維，先 x 迴圈，再 coin[i] 去加
    int n, x; cin >> n >> x;
    vector<int> coin(n + 1);
    // dp[i][j] 為考慮前 i 個硬幣，組合為 i 的組數
    vector<vector<int>> dp(2, vector<int>(x + 1, 0));
    dp[0][0] = 1;
    for (int i = 1; i <= n; i++) cin >> coin[i];
    for (int i = 1; i <= n; i++){
        for (int j = 0; j <= x; j++) {
            // 壓到 2 * n
            dp[i & 1][j] = dp[i & 1][j];
            if (j >= coin[i]) {
                (dp[i
                    & 1][j] += dp[i & 1][j - coin[i]]) %= mod;
            }
        }
    }
    cout << dp[n & 1][x];
}

void minimize_coins_nums(){
    // 有 n 種錢幣，求組合為 x 的最小硬幣數
    int n, x; cin >> n >> x;
    vector<int> coin(n);
    for (int i = 0; i < n; i++) cin >> coin[i];
    // dp[i] 是組合為 i 的最小硬幣數
    vector<int> dp(x + 1, 0);
    for (int i = 1; i <= x; i++) {
        dp[i] = 2e9;
        for(auto &j : coin){
            if(j <= i){
                dp[i] = min(dp[i], dp[i - j] + 1);
            }
        }
    }
    cout << (dp[x] == 2e9 ? -1 : dp[x]);
}

int main(){
    coin_combination_II();
    minimize_coins_nums();
}

```

## 9.4 編輯距離 [80c4dc]

```

#include <bits/stdc++.h>
using namespace std;
void solve(){
    string s1, s2; cin >> s1 >> s2;
    int size1 = s1.size(), size2 = s2.size();
    // dp[i][j] 為 s1 的前 i 個字元，跟 s2 的前 j 個字元
    vector<
        vector<int>> dp(size1 + 1, vector<int>(size2 + 1, 0));
    s1 = "0" + s1, s2 = "0" + s2;
    for (int i = 1; i <= size1; i++) dp[i][0] = i;
    for (int i = 1; i <= size2; i++) dp[0][i] = i;
    for (int i = 1; i <= size1; i++){
        for (int j = 1; j <= size2; j++) {
            if (s1[i] == s2[j]) {
                dp[i][j] = dp[i - 1][j - 1];
            }
            else {
                // s1 新增等價於 s2 砍掉
                // dp[i][j] = min(修改, s1 新增, s2 新增);
                dp[i][j] = min({dp[i - 1][
                    j - 1], dp[i - 1][j], dp[i][j - 1]} + 1;
            }
        }
    }
    cout << dp[size1][size2];
}

```

## 9.5 LCS [937a28]

```

#include <bits/stdc++.h>
using namespace std;
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('a', length);
    // along to dp to trace back
    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";
}

```

## 9.6 LIS [f23284]

```

#include <bits/stdc++.h>
using namespace std;
// Rec Sequence LIS
void solve(){
    int n; cin >> n;
    vector<int> v(n);
    for (int i = 0; i < n; i++) {
        cin >> v[i];
    }
    int dp[n]; vector<int> mono;
    mono.push_back(v[0]);
    dp[0] = 1; int L = 1;
    for (int i = 1; i < n; i++) {
        if (v[i] > mono.back()) {
            mono.push_back(v[i]);
            dp[i] = ++L;
        }
        else {
            auto it
                = lower_bound(mono.begin(), mono.end(), v[i]);
            *it = v[i];
            dp[i] = it - mono.begin() + 1;
        }
    }
    vector<int> ans;
    cout << L << endl;
    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 << " ";
    }
}

```

## 9.7 Projects [c03e88]

```

#include <bits/stdc++.h>
using namespace std;
#define all(x) (x).begin(), (x).end()
#define endl "\n"
#define int long long
const int maxn = 2e5 + 5;
struct project {
    int from, end, gain, id;
};
void solve(){
    int n; cin >> n;
    vector<project> projects(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> projects
            [i].from >> projects[i].end >> projects[i].gain;
        projects[i].id = 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, {0, 0, 0}); // nums, gain, time
    vector<int> par(n + 1, 0), ans, add(n + 1, -1);
    for (int i = 1; i <= n; i++) {
        int idx = --upper_bound(projects.begin(), projects.
            begin() + i, project({0, projects[i].from, 0, 0})),
            [(const project &a, const project &b) -> bool {
                return a.end < b.end;
            }) - projects.begin(); // 二分搜最接近 from 的 end
        // cerr << idx << "\n";
        dp[i] = dp[idx - 1];
        par[i] = i - 1;
        if (dp[i][1] < dp[idx][1] + projects[i].gain ||
            (dp[i][1]
                == dp[idx][1] + projects[i].gain && dp[i][2] > dp
                    [idx][2] + projects[i].end - projects[i].from)) {
            dp[i] = {dp[idx
                ][0] + 1, dp[idx][1] + projects[i].gain, dp
                    [idx][2] + projects[i].end - projects[i].from};
            par[i] = idx;
            add[i] = projects[i].id;
        }
    }
    cout << dp
        [n][0] << " " << dp[n][1] << " " << dp[n][2] << endl;
}

```

```

    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 << " ";
}

```

## 9.8 Removal Game [45a446]

```

#include <bits/stdc++.h>
using namespace std;
void solve(){
    int n; cin >> n;
    vector<vector<int>> dp(n + 1, vector<int>(n + 1));
    int pref = 0;
    vector<int> v(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> v[i];
        pref += v[i];
    }
    // dp[i][j] = max_diff(i to j);
    for (int i = n; i > 0; i--) {
        for (int j = 1; j <= n; j++) {
            if (i > j) continue;
            else if (i == j) {
                dp[i][j] = v[i];
            }
            else {
                dp[i][j] = max(v[i] - dp[i + 1][j], v[j] - dp[i][j - 1]); // i+1, j-1, care dp's order
            }
        }
    }
    // x + y = sum, dp[1][n] = x - y;
    cout << (pref + dp[1][n]) / 2;
}

```

```

    }
    cout << l << ' ' << u << '\n';
    return l + u;
}
int main(){
    ll n,x,y;
    cin >> n;
    for(ll i = 0; i < n; i++){
        cin >> x >> y;
        P.push_back({x,y});
    }
    ll ans = Andrew_monotone_chain(n) - 2;
    cout << ans << "\n";
    return 0;
}

```

# 10 Geometry

## 10.1 Cross Product [c37c89]

```

#include <bits/stdc++.h>
using namespace std;
const double eps = 1e-8;
struct point {
    double x, y;
    point operator * (int a){ return {a * x, a * y}; }
    point operator + (point b){ return {x + b.x, y + b.y}; }
    point operator - (point b){ return {x - b.x, y - b.y}; }
    double operator * (point b){ return x * b.x + y * b.y; }
    double operator ^ (point b){ return x * b.y - y * b.x; }
    bool operator < (point b){ return x == b.x ? y < b.y : x < b.x; }
};
double abs(point a) { return sqrt(a * a); }
int sign
    (double a) { return fabs(a) < eps ? 0 : a > 0 ? 1 : -1; }
int ori(point
    a, point b, point c) { return sign((b - a) ^ (c - a)); }
bool colinear(point a,
    point b, point c) { return sign((b - a) ^ (c - a)) == 0; }
bool between(point a, point b, point c){ // c between a and b
    if (!colinear(a, b, c)) return false;
    return sign((a - c) * (b - c)) <= 0;
}
bool intersect(point
    a, point b, point c, point d){ // line(a, b)line(c, d)
    int abc = ori(a, b, c);
    int abd = ori(a, b, d);
    int cda = ori(c, d, a);
    int cdb = ori(c, d, b);
    if(abc == 0 || abd == 0)
        return between(a, b, c) || between
            (a, b, d) || between(c, d, a) || between(c, d, b);
    return abc * abd <= 0 && cda * cdb <= 0;
}

```

## 10.2 Convex Hull [e8ad24]

```

vector<pii> P, L, U;
ll cross(pii o, pii a, pii b){ // OA OB > 0 counterclock
    return (a.first - o.first) * (b.second
        - o.second) - (a.second - o.second) * (b.first - o.first);
}
ll Andrew_monotone_chain(ll n){
    sort(P.begin(), P.end());
    ll l = 0, u = 0; // upper and lower hull
    for (ll i=0; i<n; ++i){
        while (l >= 2 && cross(L[l-2], L[l-1], P[i]) <= 0){
            l--;
            L.pop_back();
        }
        while (u >= 2 && cross(U[u-2], U[u-1], P[i]) >= 0){
            u--;
            U.pop_back();
        }
        l++;
        u++;
        L.push_back(P[i]);
        U.push_back(P[i]);
    }
}

```