1 Data Structure

1.1 CDQ

```cpp
#define fastio ios_base::sync_with_stdio(false); cin.tie(0);
#define endl '\n'
#define _ << ' ' <<
const int INF = 2e9;
struct info {
    int t, p;
    long long x;
    // 當 x 界在 -1e9 ~ 1e9，代表這是修改
    // 當 x >= INF，代表這是詢問
    // 並且詢問的編號為 (x - INF)
    // 例：第 2 次詢問，那 x 就設為 INF + 2
    // 當 x <= -INF，一樣代表詢問，只不過是倒扣的詢問
};
int n, q;
vector<info> v;
vector<long long> ans;
void cdq(int l, int r) {
    if (l == r) return;
    int mid = (l+r) / 2;
    cdq(l, mid);
    cdq(mid+1, r);
    vector<info> temp;
    long long add_sum = 0;
    int pl = l, pr = mid+1;
    while (pl <= mid && pr <= r) {
        if (v[pl].p <= v[pr].p) {
            temp.emplace_back(v[pl]);
            if (abs(v[pl].x) <= 1e9) {
                add_sum += v[pl].x;
            }
            pl++;
        }
        else {
            temp.emplace_back(v[pr]);
            if (v[pr].x >= INF) {
                ans[v[pr].x - INF] += add_sum;
            } else if (v[pr].x <= -INF) {
                ans[-(v[pr].x + INF)] -= add_sum;
            }
            pr++;
        }
    }
    while (pl <= mid) {
        temp.emplace_back(v[pl]);
        pl++;
    }
    while (pr <= r) {
        temp.emplace_back(v[pr]);
        if (v[pr].x >= INF) ans[v[pr].x - INF] += add_sum;
        else if (v[pr].x <= -INF) ans[-(v[pr].x + INF)] -= add_sum;
        pr++;
    }
    for (int i = l, j = 0; i <= r; i++, j++) v[i] = temp[j];
}

int main() {
    fastio;
    cin >> n >> q;
    v.clear(); ans.clear();
    int time = 0;
    vector<long long> arr(n+1);
    for (int p = 1; p <= n; p++) {
        long long x; cin >> x;
        v.emplace_back(info{++time, p, x});
        arr[p] = x;
    }
    int ask_id = 0;
    for (int i = 0; i < q; i++) {
        int op; cin >> op;
        if (op == 1) {
            int p; long long x;
            cin >> p >> x;
            v.emplace_back(info{++time, p, x - arr[p]});
            arr[p] = x;
        }
        else {
            int l, r; cin >> l >> r;
            v.emplace_back(info{++time, r, INF + ask_id});
            if (l > 1) v.emplace_back(info{time, l-1, -INF - ask_id});
            ask_id++;
            ans.emplace_back(0);
        }
    }
    cdq(0, (int)v.size()-1);
    for (int i = 0; i < (int)ans.size(); i++)
        cout << ans[i] << endl;
}

struct info {
    int x, y, z, id; // id 是數對在原本陣列的位置
};
int N;
vector<info> v;
vector<int> ans;
struct BIT {
    vector<int> bit;
    int n;
    BIT(int n) : n(n) bit.resize(n + 1, 0);
    void upd(int idx, int val) {
        while (idx <= n) {
            bit[idx] += val;
            idx += idx & -idx;
        }
    }
    int qry(int idx) {
        int sum = 0;
        while (idx > 0) {
            sum += bit[idx];
            idx -= idx & -idx;
        }
        return sum;
    }
};

void cdq(int L, int R, BIT& BITree) {
    if (L == R) return;
    int mid = (L + R) / 2;
    cdq(L, mid, BITree);
    cdq(mid + 1, R, BITree);
    vector<info> temp;
    vector<pair<int, int>> BIT_op; // BIT_op: 紀錄你在BIT的操作,包含修改位置、修改的值
    int pl = L, pr = mid + 1;
    while (pl <= mid && pr <= R) {
        if (v[pl].y < v[pr].y) {
            temp.emplace_back(v[pl]);
            BITree.upd(v[pl].z, 1);
            BIT_op.emplace_back(make_pair(v[pl].z, 1));
            pl++;
        } else {
            temp.emplace_back(v[pr]);
            ans[v[pr].id] += BITree.qry(v[pr].z - 1);
            pr++;
        }
    }
    while (pl <= mid) temp.emplace_back(v[pl]), pl++;
    while (pr <= R) {
        temp.emplace_back(v[pr]);
        ans[v[pr].id] += BITree.qry(v[pr].z - 1);
        pr++;
    }
    for (int pt = 0, pv = L; pv <= R; pt++, pv++) v[pv] = temp[pt];
    for (auto& op : BIT_op) {
        BITree.upd(op.first, -op.second);
    }
}
int main() {
    fastio;
    cin >> N;
    v.resize(N);
    ans.resize(N, 0);
    for (int i = 0; i < N; i++) {
        cin >> v[i].x >> v[i].y >> v[i].z, v[i].id = i;
    sort(v.begin(), v.end(), [](const info& a, const info& b) {return a.x < b.x;});
    BIT BITree(N);
    cdq(0, N - 1, BITree);
    for (int i = 0; i < N; i++) cout << ans[i] << endl;
}
```

1.2 DSU

```cpp
class UnionFind {
private:
    vector<int> parent;
    vector<int> rank;
public:
    UnionFind(int size) : parent(size), rank(size, 1) {
        for (int i = 0; i < size; ++i) {
            parent[i] = i;
        }
    }
    int getSize() const return parent.size();
    int find(int p) {
        if (p < 0 || p >= parent.size()) throw out_of_range("Out of bound.");
        while (p != parent[p]) {
            parent[p] = parent[parent[p]];
            p = parent[p];
        }
        return p;
    }
    int findR(int p) {
        if (p < 0 || p >= parent.size())
            throw out_of_range("Out of bound.");
        if (p != parent[p])
```

```cpp
            parent[p] = findR(parent[p]);
        return parent[p];
    }
    bool isConnected(int p, int q){ return find(p) == find(q);}
    void unionElements(int p, int q) {
        int pRoot = find(p);
        int qRoot = find(q);
        if (pRoot == qRoot) return;
        if (rank[pRoot] < rank[qRoot]) parent[pRoot] = qRoot;
        else if (rank[qRoot] < rank[pRoot]) parent[qRoot] = pRoot;
        else {
            parent[pRoot] = qRoot;
            rank[qRoot]++;
        }
    }

    int countConnectedNodes(int p) {
        int root = find(p);
        int count = 0;
        for (int i = 0; i < parent.size(); i++) {
            if (find(i) == root) count++;
        }
        return count;
    }
};
int main() {
    UnionFind uf(10);
    uf.unionElements(1, 2);
    uf.unionElements(2, 3);
    uf.unionElements(4, 5);
    uf.unionElements(3, 4);
}
```

1.3 Dag

```cpp
void DFS(map<int, queue<int>>& graph, map<int, bool>& visited, int place, stack<int>& ans) {
    if (visited[place]) return;
    visited[place] = true;
    while (!graph[place].empty()) {
        DFS(graph, visited, graph[place].front(), ans);
        graph[place].pop();
    }
    ans.push(place);
}
```

1.4 Kruskal

```cpp
struct Edge {
    int src, dest, weight;
    Edge(int s, int d, int w) : src(s), dest(d), weight(w) {}
};
int findParent(const vector<int>& parent, int i) {
    return parent[i] == i ? i : findParent(parent, parent[i]);
}
vector<Edge> kruskalMST(const vector<vector<int>>& adjacencyMatrix) {
    int size = adjacencyMatrix.size();
    vector<Edge> mst, edges;
    for(int i = 0; i < size; ++i){
        for(int j = i + 1; j < size; ++j){
            if(adjacencyMatrix[i][j] > 0){
                edges.push_back(Edge(i, j, adjacencyMatrix[i][j]));
            }
        }
    }
    sort(edges.begin(), edges.end(), [](const Edge& e1, const Edge& e2) {
        return e1.weight < e2.weight;
    });
    vector<int> parent(size);
    for(int i = 0; i < size; ++i) parent[i] = i;
    int count = 0;
    for(const Edge& edge : edges) {
        if(count == size - 1) break;
        int srcParent = findParent(parent, edge.src), destParent = findParent(parent, edge.dest);
        if(srcParent != destParent) {
            mst.push_back(edge);
            ++count;
            parent[srcParent] = destParent;
        }
    }
    return mst;
}
```

1.5 Prim

```cpp
struct Edge {
    int src, dest, weight;
    Edge(int s, int d, int w) : src(s), dest(d), weight(w) {}
};
vector<Edge> primMST(const vector<vector<int>>& adjacencyMatrix) {
    int size = adjacencyMatrix.size();
    vector<bool> visited(size, false);
    vector<Edge> mst;
    priority_queue<Edge, vector<Edge>, function<bool(Edge, Edge)>> pq(
        [](const Edge& e1, const Edge& e2) { return e1.weight > e2.weight; }
    );
    visited[0] = true;
    for (int i = 1; i < size; ++i) {
        if (adjacencyMatrix[0][i] > 0) {
            pq.push(Edge(0, i, adjacencyMatrix[0][i]));
        }
    }
    while (!pq.empty()) {
        Edge minEdge = pq.top();
        pq.pop();
        int u = minEdge.dest;
        if (visited[u]) continue;
        visited[u] = true;
        mst.push_back(minEdge);
        for (int v = 0; v < size; ++v) {
            if (adjacencyMatrix[u][v] > 0 && !visited[v]) {
                pq.push(Edge(u, v, adjacencyMatrix[u][v]));
            }
        }
    }
    return mst;
}

int main() {
    // Example adjacency matrix
    vector<vector<int>> adjacencyMatrix = {
        {0, 2, 0, 6, 0},
        {2, 0, 3, 8, 5},
        {0, 3, 0, 0, 7},
        {6, 8, 0, 0, 9},
        {0, 5, 7, 9, 0}
    };
    vector<Edge> mst = primMST(adjacencyMatrix);
    cout << "Minimum Spanning Tree Edges:\n";
    for (const auto& edge : mst) {
        cout << "Src: " << edge.src << ", Dest: " << edge.dest << ", Weight: " << edge.weight << "\n";
    }

    return 0;
}
```

1.6 SegmentTreeLazy

```cpp
#define Ls(x) x << 1
#define Rs(x) x << 1 | 1
int N = 1e5+5;
vector<int> segTree(4*N), lazy(4*N) , arr(N);
void build(int l, int r, int idx = 1){
    if(l == r){
        segTree[idx] = arr[l];
        return;
    }
    int mid = (l+r) >> 1;
    build(l, mid, Ls(idx));
    build(mid+1, r, Rs(idx));
    segTree[idx] = segTree[Ls(idx)]+ segTree[Rs(idx)];
}
void pushDown(int l, int r, int idx){
    if(lazy[idx] == 0) return;
    segTree[idx] += (r-l+1)*lazy[idx];
    if(l != r) lazy[Ls(idx)] += lazy[idx],lazy[Rs(idx)] += lazy[idx];
    lazy[idx] = 0;
}
void update(int l, int r, int ql, int qr, int val, int idx = 1){
    pushDown(l, r, idx);
    if(l > qr || r < ql) return;
    if(l >= ql && r <= qr){
        segTree[idx] += (r-l+1)*val;
        if(l != r) lazy[Ls(idx)] += val, lazy[Rs(idx)] += val;
        return;
    }
    int mid = (l+r) >> 1;
    update(l, mid, ql, qr, val, Ls(idx));
    update(mid+1, r, ql, qr, val, Rs(idx));
    segTree[idx] = segTree[Ls(idx)]+ segTree[Rs(idx)];
}

int query(int l, int r, int ql, int qr, int idx = 1){
    pushDown(l, r, idx);
    if(l > qr || r < ql) return 0;
    if(l >= ql && r <= qr) return segTree[idx];
    int mid = (l+r) >> 1, sum = 0;
    if(ql <= mid) sum += query(l, mid, ql, qr, Ls(idx));
    if(qr > mid) sum += query(mid+1, r, ql, qr, Rs(idx));
    return sum;
}
```

```cpp
int32_t main(){
    int n, t;
    cin >> n >> t;
    for(int i = 1; i <= n; i++) cin >> arr[i];
    build(1, n, 1);
    for(int i = 0; i < t; i++){
        int j, l, r, v;
        cin >> j;
        if(j == 1){
            cin >> l >> r >> v;
            update(1, n, l, r, v);
        } else {
            cin >> r;
            cout << query(r, 1, n) << endl;
        }
    }
}
// prefix sum
int N = 2e5 + 5;
vector<ll> segTree(4 * N, -2e18), lazy(4 * N), arr(N), tmp(N);
void build(int l, int r, int idx = 1){
    if(l == r){
        segTree[idx] = arr[l];
        return;
    }
    int mid = (l + r) >> 1;
    build(l, mid, Ls(idx));
    build(mid + 1, r, Rs(idx));
    segTree[idx] = max(segTree[Ls(idx)] + segTree[Rs(idx)]);
}
void pushDown(int l, int r, int idx, int x = 0){
    if(lazy[idx] == 0) return;
    segTree[idx] += lazy[idx];
    int mid = (l + r) / 2;
    if(l != r) lazy[Ls(idx)] += lazy[idx], lazy[Rs(idx)] += lazy[idx];
    lazy[idx] = 0;
}
void update(int l, int r, int ql, int qr, int val, int idx = 1){
    pushDown(l, r, idx);
    if(l > qr || r < ql) return;
    if(l >= ql && r <= qr){
        segTree[idx] += val;
        pushDown(l, r, idx);
        return;
    }
    int mid = (l + r) >> 1;
    update(l, mid, ql, qr, val, Ls(idx));
    update(mid + 1, r, ql, qr, val, Rs(idx));
    segTree[idx] = max(segTree[Ls(idx)], segTree[Rs(idx)]);
}
ll query(int l, int r, int ql, int qr, int idx = 1){
    pushDown(l, r, idx);
    if(l > qr || r < ql) return -2e18;
    if(l >= ql && r <= qr) return segTree[idx];
    int mid = (l + r) >> 1, ans = -2e18;
    ans = max(ans, query(l, mid, ql, qr, Ls(idx)));
    ans = max(ans, query(mid + 1, r, ql, qr, Rs(idx)));
    segTree[idx] = max(segTree[Ls(idx)], segTree[Rs(idx)]);
    return ans;
}
int main(){
    IO;
    int n, q;
    cin >> n >> q;
    for(int i = 1; i <= n; i++) cin >> tmp[i];
    arr = tmp;
    for(int i = 2; i <= n; i++) arr[i] += arr[i - 1];
    build(1, n, 1);
    while(q--){
        int j, k, u;
        cin >> j >> k >> u;
        if(j == 1){
            update(1, n, k, n, u - tmp[k]), tmp[k] = u;
        } else {
            cout << max(0ll, query(1, n, k, u) - (k!=1?query(1, n, k - 1, k - 1):0)) << endl;
        }
    }
}
```

1.7 SegmentTree

```cpp
ll v[1e5] , seg[4e5+7] ;
void StructSEG(int l , int r , ll node){
    if(l==r) seg[node] = v[r] ;
    else{
        int mid = (l+r) >> 1 , tmp = node << 1;
        StructSEG(l,mid,tmp) ;
        StructSEG(mid+1,r,tmp+1) ;
        seg[node] = seg[tmp] + seg[tmp+1] ;
    }
}
int query(int tL , int tR , int nL , int nR , int node){
    int mid = (nL+nR)>>1 , ans = 0 ;
    if(tL <= nL && nR <= tR) return seg[node] ;
    if(tL <= mid) ans += query(tL,tR,nL   ,mid,(node<<1)  );
    if(tR  > mid) ans += query(tL,tR,mid+1,nR ,(node<<1)+1);
    return ans ;
}
void updateNode(int idx , int val , int l , int r , int node) {
    if(l == r) seg[node] = val , v[idx] = val ;
    else{
        int m = (l + r) >> 1 ;
        int leftNode  = (node << 1) ;
        int rightNode = (node << 1) + 1 ;
        if(idx <= m && idx >= l) updateNode(idx , val , l , m , leftNode ) ;
        else updateNode(idx , val , m+1,r , rightNode ) ;
        seg[node] = seg[leftNode] + seg[rightNode] ;
    }
}
```

2 DP

2.1 BellmanFord

```cpp
typedef pair<int, int> pii;
const int maxn = 1e5 + 5;
const int INF = 0x3f3f3f3f;
vector<pii> G[maxn];
int dis[maxn];
bool BellmanFord(int n, int s) {
    for (int i = 1; i <= n; i++) dis[i] = INF;
    dis[s] = 0;
    bool relax;
    for (int r = 1; r <= n; r++) {
        relax = false;
        for (int i = 1; i <= n; i++) {
            for (pii e : G[i]) {
                if (dis[i] + e.second < dis[e.first]) {
                    dis[e.first] = dis[i] + e.second;
                    relax = true;
                }
            }
        }
        if (!relax) break; // Optimization: If no relaxation happened, break early
    }
    return relax; // If relax is true after n iterations, there is a negative cycle
}
int main() {
    int n, m, s;
    cin >> n >> m >> s;
    while (m--) {
        int u, v, w;
        cin >> u >> v >> w;
        G[u].emplace_back(v, w);
    }
    if (BellmanFord(n, s)) {
        cout << "negative cycle\n";
    } else {
        for (int i = 1; i <= n; i++) {
            if (dis[i] == INF) {
                cout << "INF\n"; // Print INF if there's no path
            } else {
                cout << dis[i] << " \n"[i == n];
            }
        }
    }
}
```

2.2 EdmondsKarp

```cpp
int main() {
    int n, m, b, e, s, t, w, q = 1;
    while (cin >> n, n) {
        int ans = 0, go = 1, Min;
        vector<vector<int>> a(n+1, vector<int>(n+1, 0));
        vector<int> last(n+1, 0), can(n+1, 0);
        cin >> b >> e >> m;
        while (m--) {
            cin >> s >> t >> w;
            a[s][t] = a[t][s] += w;
        }
        while (go) {
            go = 0;
            queue<int> Q;
            Q.push(b);
            Min = 99999999;
            fill(can.begin(), can.end(), 0);
            while (Q.size()) {
                int top = Q.front();
                Q.pop();
                can[top] = 1;
                for (int k = 1; k <= n; k++) {
                    if (a[top][k] && !can[k]) {
                        last[k] = top;
                        if (k == e) {
                            go = 1;
                            break;
                        }
                        Q.push(k);
                    }
                }
                if (go) {
                    int S = s = e;
                    while (last[s]) {
                        Min = min(Min, a[last[s]][s]);
                        s = last[s];
                    }
                    while (last[S]) {
                        a[last[S]][S] -= Min;
                        a[S][last[S]] += Min;
                        S = last[S];
                    }
                    ans += Min;
                    fill(last.begin(), last.end(), 0);
                    fill(can.begin(), can.end(), 0);
                    break;
                }
            }
        }
        printf("Network %d\nThe bandwidth is %d.\n\n", q++, ans);
    }
}
```

2.3 LCS

```
若 一 樣 : 左 上 +1
else max(左 , 上 )
```

2.4 LCS2LIS

```
int LCS2LIS(string &a, string &b) {
    vector<int> List;
    for (int i = 0; i < b.size(); i++) {
        for (int j = 0; j < a.size(); j++) if (b[i] == a[j]) List.push_back(j);
    }
    if (List.size() == 0) return 0;
    vector<int> dp;
    for (int i = 0; i < List.size(); i++) {
        if (dp.empty() || List[i] > dp.back()) dp.push_back(List[i]);
        else *lower_bound(dp.begin(), dp.end(), List[i]) = List[i];
    }
    return dp.size();
}
```

2.5 LCS3D

```
// space O(m*n*o) version
int lcsOf3(string X, string Y, string Z, int m, int n, int o) {
    int L[m+1][n+1][o+1];
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            for (int k = 0; k <= o; k++) {
                if (i == 0 || j == 0 || k == 0) {
                    L[i][j][k] = 0;
                } else if (X[i-1] == Y[j-1] && X[i-1] == Z[k-1]) {
                    L[i][j][k] = L[i-1][j-1][k-1] + 1;
                } else {
                    L[i][j][k] = max({L[i-1][j][k], L[i][j-1][k], L[i][j][k-1]});
                }
            }
        }
    }
    return L[m][n][o];
}

// space O(2*n*o) version
int lcsOf3(string X, string Y, string Z, int m, int n, int o) {
    vector<vector<vector<int>>> L(2, vector<vector<int>>(n+1, vector<int>(o+1, 0)));
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            for (int k = 0; k <= o; k++) {
                if (i == 0 || j == 0 || k == 0) {
                    L[i % 2][j][k] = 0;
                } else if (X[i-1] == Y[j-1] && X[i-1] == Z[k-1]) {
                    L[i % 2][j][k] = L[(i-1) % 2][j-1][k-1] + 1;
                } else {
                    L[i % 2][j][k] = max({L[(i-1) % 2][j][k], L[i % 2][j-1][k], L[i % 2][j][k-1]});
                }
            }
        }
    }
    return L[m % 2][n][o];
}
```

2.6 LIS

```
void LIS(vector<int> &arr) {
    vector<int> lis;
    for (int i = 0; i < arr.size(); i++) {
        auto it = lower_bound(lis.begin(), lis.end(), arr[i]);
        if (it == lis.end()) lis.push_back(arr[i]);
        else *it = arr[i];
    }
    cout << lis.size() << endl;
}
```

2.7 MaxFlow

```
struct edge {
    int from, to, cap, flow;
    edge(int u, int v, int c, int f) : from(u), to(v), cap(c), flow(f) {}
};
struct Dinic {
    vector<edge> edges;
    vector<vector<int>> adj;
    vector<int> level, ptr;
    int n, m = 0, s, t;
    Dinic(int n, int s, int t) : n(n), s(s), t(t) {
        adj.resize(n);
        level.resize(n);
        ptr.resize(n);
    }
    void addEdge(int u, int v, int c) {
        edges.emplace_back(u, v, c, 0);
        edges.emplace_back(v, u, 0, 0);
        adj[u].push_back(m);
        adj[v].push_back(m + 1);
        m += 2;
    }
    bool bfs() {
        fill(level.begin(), level.end(), -1);
        queue<int> q;
        q.push(s);
        level[s] = 0;
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for (int id : adj[u]) {
                if (edges[id].cap - edges[id].flow < 1) continue;
                int v = edges[id].to;
                if (level[v] != -1) continue;
                level[v] = level[u] + 1;
                q.push(v);
            }
        }
        return level[t] != -1;
    }
    int dfs(int u, int pushed) {
        if (pushed == 0) return 0;
        if (u == t) return pushed;
        for (int& cid = ptr[u]; cid < (int)adj[u].size(); cid++) {
            int id = adj[u][cid];
            int v = edges[id].to;
            if (level[u] + 1 != level[v] || edges[id].cap - edges[id].flow < 1) continue;
            int tr = dfs(v, min(pushed, edges[id].cap - edges[id].flow));
            if (tr == 0) continue;
            edges[id].flow += tr;
            edges[id ^ 1].flow -= tr;
            return tr;
        }
        return 0;
    }
    int flow() {
        int f = 0;
        while (true) {
            if (!bfs()) break;
            fill(ptr.begin(), ptr.end(), 0);
            while (int pushed = dfs(s, INT_MAX)) f += pushed;
        }
        return f;
    }
};
int main() {
    int n, m, s, t;
    cin >> n >> m >> s >> t;
    Dinic dinic(n, s, t);
    for (int i = 0; i < m; i++) {
        int u, v, c;
        cin >> u >> v >> c;
        dinic.addEdge(u, v, c);
    }
    cout << dinic.flow() << endl;
}
```

2.8 SPFA

```
typedef pair<int, int> pii;
const int maxn = 1e5 + 5;
const int INF = 0x3f3f3f3f;
vector<pii> G[maxn];
int dis[maxn];

void SPFA(int n, int s) {
    for (int i = 1; i <= n; i++) dis[i] = INF;
    dis[s] = 0;
    queue<int> q;
    q.push(s);
    bool inque[maxn] = {};
    inque[s] = true;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        inque[u] = false;
        for (pii e : G[u]) {
            int v = e.first, w = e.second;
            if (dis[u] + w < dis[v]) {
                dis[v] = dis[u] + w;
                if (!inque[v]) {
                    q.push(v);
                    inque[v] = true;
                }
            }
        }
    }
}
int main() {
    int n, m, s;
    cin >> n >> m >> s;
    while (m--) {
        int u, v, w;
        cin >> u >> v >> w;
        G[u].emplace_back(v, w);
    }
    SPFA(n, s);
    for (int i = 1; i <= n; i++) {
        if (dis[i] == INF) cout << "INF ";
        else cout << dis[i] << " ";
    }
}
```

2.9 DP

```
#include <iostream>
#define N 205
#define MAX 0x03fffffff
using namespace std;

int Metrix[N][N], dist[N], vist[N], path[N];
struct edge { int s, t, cost; } E[N];

void OutPath(int s, int t, int NV) {
    for (int i = 1; i <= NV; i++) cout << path[i] << " ";
    cout << endl;
    int u = s, v = t;
    while (v != s) {
        cout << v << "-->";
        v = path[v];
    }
    cout << u << endl;
}

int Dijkstra(int s, int t, int NV) {
    for (int i = 1; i <= NV; i++) {
        dist[i] = MAX;
        vist[i] = 0;
        path[i] = i;
    }
    dist[s] = 0;
    for (int i = 1; i <= NV; i++) {
        int min_value = MAX, u = -1;
        for (int j = 1; j <= NV; j++)
            if (!vist[j] && dist[j] < min_value)
                min_value = dist[j], u = j;
        if (u == -1) break;
        vist[u] = 1;
        for (int j = 1; j <= NV; j++)
            if (!vist[j] && dist[u] + Metrix[u][j] < dist[j])
                dist[j] = dist[u] + Metrix[u][j], path[j] = u
                    ;
    }
    return dist[t] == MAX ? -1 : dist[t];
}

void SP2th(int s, int t, int NV) {
    int flag = Dijkstra(s, t, NV);
    if (flag == -1) {
        cout << "不可达" << endl;
        return;
    }
    int u = s, v = t, arcNum = 0;
    while (v != u) {
        cout << v << "-->";
        E[arcNum++] = {v, path[v], Metrix[v][path[v]]};
        v = path[v];
    }
    cout << u << endl << ":" << dist[t] << endl;
    int min_SP = MAX;
    for (int i = 0; i < arcNum; i++) {
        u = E[i].s; v = E[i].t;
        Metrix[u][v] = Metrix[v][u] = MAX;
        flag = Dijkstra(s, t, NV);
        if (flag != -1) {
            OutPath(s, t, NV);
            cout << ":" << dist[t] << endl;
            if (min_SP > dist[t]) min_SP = dist[t];
        }
        Metrix[u][v] = Metrix[v][u] = E[i].cost;
    }
    cout << "次短路：" << min_SP << endl;
}

int main() {
    int m, n, s, t, c;
    while (cin >> n >> m) {
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                Metrix[i][j] = (i == j) ? 0 : MAX;
        for (int i = 0; i < m; i++) {
            cin >> s >> t >> c;
            Metrix[s][t] = Metrix[t][s] = min(Metrix[t][s], c
                );
        }
        cin >> s >> t;
        SP2th(s, t, n); //求s->t的次短路
    }
}
```

3 Geometry

3.1 ClosestPair

```
typedef pair<ll, ll> pii;
#define x first
#define y second
ll dd(const pii& a, const pii& b) {
    ll dx = a.x - b.x, dy = a.y - b.y;
    return dx * dx + dy * dy;
}
const ll inf = 1e18;
ll dac(vector<pii>& p, int l, int r) {
    if (l >= r) return inf;
    int m = (l + r) / 2;
    ll d = min(dac(p, l, m), dac(p, m + 1, r));
    vector<pii> t;
    for (int i = m; i >= l && p[m].x - p[i].x < d; i--)
        t.push_back(p[i]);
    for (int i = m + 1; i <= r && p[i].x - p[m].x < d; i++)
        t.push_back(p[i]);
    sort(t.begin(), t.end(),
         [](pii& a, pii& b) { return a.y < b.y; });
    int n = t.size();
    for (int i = 0; i < n - 1; i++)
        for (int j = 1; j < 4 && i + j < n; j++)
            // 這裡可以知道是哪兩點是最小點對
            d = min(d, dd(t[i], t[i + j]));
    return d;
}
// 給一堆點，求最近點對的距離「的平方」。
ll closest_pair(vector<pii>& pp) {
    sort(pp.begin(), pp.end());
    return dac(pp, 0, pp.size() - 1);
}
```

3.2 ConvexHull

```
void Dhull(vector<pair<double,double>>points,vector<pair<
    double,double>>&hull,int e){
    if(e)hull.push_back(points[0]);hull.push_back(points[1]);
    for(int i=2;i<points.size();i++){
        while(hull.size()>=2){
            pair<double,double>p1=hull[hull.size()-2],p2=hull
                [hull.size()-1],p3=points[i];
            double x1=p2.first-p1.first,y1=p2.second-p1.
                second;
            double x2=p3.first-p2.first,y2=p3.second-p2.
                second;
            if(x1*y2-x2*y1<=0)break;
            hull.pop_back();
        }
        hull.push_back(points[i]);
    }
}
int main(){
    string s,ss;
    while(getline(cin,s)){
        double n,x,y,t=0;char c;
        vector<pair<double,double>>points,hull;
        istringstream sin(s);
        while(sin>>c>>x>>c>>y>>c){
            points.push_back({x,y});
        }
        sort(points.begin(),points.end(),[](auto a,auto b){if(a.second==b.second)return a.first<b.first; return a.second<b.second;});
        Dhull(points,hull,1);
        reverse(points.begin(),points.end());
        Dhull(points,hull,0);
        reverse(hull.begin(),hull.end());
        for(auto&k:hull){
            if(t++)cout<<' ';
            cout<<'('<<k.first<<','<<k.second<<')';
        }
    }
}
```

3.3 EulerCircuit

```
1.無向圖:如果恰有兩點的度數為奇數,則存在歐拉路徑,此二點分別
為起終點;如果全部的點度數都是偶數,則存在歐拉迴路。
2.有向圖:如果恰有一點的出度等於入度+1、另有一點的入度
等於出度+1,其餘皆入度等於出度,則存在歐拉路徑,此二點
分別為起終點;如果全部的點入度等於出度,則存在歐拉迴
路。
```

3.4 Hopcroft-Karp

```
const int maxn = 1000; // Example maximum number of nodes, adjust as needed
int n, m, vis[maxn], level[maxn], pr[maxn], pr2[maxn];
vector<int> edge[maxn]; // Adjacency list for the left partition

bool dfs(int u) {
    vis[u] = true;
    for (int v : edge[u]) {
        int pv = pr2[v];
        if (pv == -1 || (!vis[pv] && level[u] < level[pv] && dfs(pv))) {
            pr[u] = v;
            pr2[v] = u;
            return true;
        }
    }
    return false;
}

int hopcroftKarp() {
    memset(pr, -1, sizeof(pr));
    memset(pr2, -1, sizeof(pr2));
    int match = 0;
    while (true) {
        queue<int> Q;
        for (int i = 1; i <= n; ++i) {
            if (pr[i] == -1) {
                level[i] = 0;
                Q.push(i);
            } else level[i] = -1;
        }
        while (!Q.empty()) {
            int u = Q.front(); Q.pop();
            for (int v : edge[u]) {
                int pv = pr2[v];
                if (pv != -1 && level[pv] < 0) {
                    level[pv] = level[u] + 1;
                    Q.push(pv);
                }
            }
        }
        memset(vis, 0, sizeof(vis));
        int d = 0;
        for (int i = 1; i <= n; ++i) if (pr[i] == -1 && dfs(i)) ++d;
        if (d == 0) return match;
        match += d;
    }
}

int main() {
    cin >> n >> m; // n: number of nodes in the left partition, m: number of nodes in the right partition
    int edges;
    cin >> edges;
    for (int i = 0; i < edges; i++) {
        int u, v;
        cin >> u >> v;
        edge[u].push_back(v + n); // Shift right partition indices by n
        edge[v + n].push_back(u); // Add reverse edge for symmetry
    }
    cout << "Maximum number of matches: " << hopcroftKarp() << endl;
}
```

3.5 Hungarian

```
const int INF = 2e9;
const int N = 1000; // Example maximum number of nodes, adjust as needed
int vis[N], m[N];
vector<int> g[N]; // Adjacency list for the bipartite graph

int dfs(int s) {
    for (int x : g[s]) {
        if (vis[x]) continue;
        vis[x] = 1;
        if (m[x] == -1 || dfs(m[x])) {
            m[x] = s;
            m[s] = x;
            return 1;
        }
    }
    return 0;
}
int hungarian(int p) { // p: number of women
    memset(m, -1, sizeof(m));
    int c = 0;
    for (int i = 0; i < p; i++) {
        if (m[i] == -1) {
            memset(vis, 0, sizeof(vis));
            c += dfs(i);
        }
    }
    return c; // Number of successful matches
}
int main() {
    int p, q, edges;
    cin >> p >> q >> edges; // p: number of women, q: number of men, edges: number of edges
    for (int i = 0; i < edges; i++) {
        int u, v;
        cin >> u >> v;
        g[u].push_back(v + p); // Shift men's indices by p
        g[v + p].push_back(u);
    }
    cout << "Maximum number of matches: " << hungarian(p) << endl;
}
```

3.6 MinCircle

```
using PT = point<T>;
using CPT = const PT;
PT circumcenter(CPT &a, CPT &b, CPT &c) {
  PT u = b-a, v =c-a;
  T c1 = u.abs2()/2, c2 = v.abs2()/2;
  T d = u.cross(v);
  return PT(a.x+(v.y*c1-u.y*c2)/d, a.y+(u.x*c2-v.x*c1)/d);
}
void solve(PT p[], int n, PT &c, T &r2){
  random_shuffle(p,p+n);
  c = p[0]; r2 = 0; // c,r2 = 圓心,半徑平方
  for(int i=1; i<n; i++)
    if( (p[i]-c).abs2() > r2) {
      c=p[i]; r2=0;
      for(int j=0; j<i; j++)
        if( (p[j]-c).abs2() > r2) {
          c.x = (p[i].x+p[j].x)/2;
          c.y = (p[i].y+p[j].y)/2;
          r2 = (p[j]-c).abs2();
          for(int k=0; k<j; k++)
            if( (p[k]-c).abs2() > r2) {
              c = circumcenter(p[i], p[j], p[k]);
              r2 = (p[i]-c).abs2();
            }
        }
    }
}
```

3.7 Angle

```
// 計算每個內角角度
// 有序的點集合，計算每個內角角度
// 逆時針方向，從第一個點開始
// 0 <= angle <= 2*PI
vector<double> angle(const vector<pair<int, int>>& v) {
    vector<double> ret;
    for (int i = 0; i < v.size(); i++) {
        int x1 = v[(i - 1 + v.size()) % v.size()].first - v[i].first;
        int y1 = v[(i - 1 + v.size()) % v.size()].second - v[i].second;
        int x2 = v[(i + 1) % v.size()].first - v[i].first;
        int y2 = v[(i + 1) % v.size()].second - v[i].second;
        double a = atan2(y1, x1) - atan2(y2, x2);
        if (a < 0) a += 2 * M_PI;
        ret.push_back(a);
        // ret = ret*180/M_PI // convert to degree
    }
    return ret;
}

// 計算三個點的內角角度
// 0 <= angle <= 2*PI
double angle(const pair<int, int>& a, const pair<int, int>& b
    , const pair<int, int>& c) {
    int x1 = a.first - b.first;
    int y1 = a.second - b.second;
    int x2 = c.first - b.first;
    int y2 = c.second - b.second;
    double ret = atan2(y1, x1) - atan2(y2, x2);
    if (ret < 0) ret += 2 * M_PI;
    return ret;
    // ret = ret*180/M_PI // convert to degree
}

// 求兩線交點 (兩線必相交)
// 0 <= angle <= 2*PI
pair<double, double> intersection(const pair<double, double>&
      p1, const pair<double, double>& p2, const pair<double,
     double>& q1, const pair<double, double>& q2) {
    double a1 = p2.second - p1.second;
    double b1 = p1.first - p2.first;
    double c1 = a1 * p1.first + b1 * p1.second;
    double a2 = q2.second - q1.second;
    double b2 = q1.first - q2.first;
    double c2 = a2 * q1.first + b2 * q1.second;
    double det = a1 * b2 - a2 * b1;
    return make_pair((b2 * c1 - b1 * c2) / det, (a1 * c2 - a2
         * c1) / det);
}
```

3.8 旋轉卡尺

```
typedef pair<ll, ll> pii;
#define x first
#define y second
#define ii (i + 1) % n  // 打字加速!
inline pii operator-(const pii& a, const pii& b) {
    return {a.x - b.x, a.y - b.y};
}  // const 不可省略
inline ll operator*(const pii& a, const pii& b) {
    return a.x * b.y - a.y * b.x;
}
inline ll crzf(const pii& o, const pii& a, const pii& b) {
    return (a - o) * (b - o)
}
inline ll dd(const pii& a, const pii& b) {
    ll dx = a.x - b.x, dy = a.y - b.y;
    return dx * dx + dy * dy;
}
// 給平面上任意個點，求其凸包。返回順序為逆時針。此方法會移除
     重複點。
#define jud \
    crzf(ret[ret.size() - 2], ret.back(), pp[i]) <= 0
vector<pii> makepoly(vector<pii>& pp) {
    int n = pp.size();
    sort(pp.begin(), pp.end());
    pp.erase(unique(pp.begin(), pp.end()), pp.end());
    vector<pii> ret;
    for (int i = 0; i < n; i++) {
        while (ret.size() >= 2 && jud) ret.pop_back();
        ret.push_back(pp[i]);
    }
    for (int i = n - 2, t = ret.size() + 1; i >= 0; i--) {
        while (ret.size() >= t && jud) ret.pop_back();
        ret.push_back(pp[i]);
    }
    if (n >= 2) ret.pop_back();
    return ret;
}
// (shoelace formula)
// 給凸包，問其面積「的兩倍」。若凸包少於三個點，回傳零。
ll area(vector<pii>& poly) {
    int n = poly.size();
    ll ret = 0;
    for (int i = 0; i < n; i++)
        ret += (poly[i].x * poly[ii].y);
    for (int i = 0; i < n; i++)
        ret -= (poly[i].y * poly[ii].x);
    return ret;
}
// 給凸包，問其兩點最遠距離「的平方」。若要問平面上任意個點的
     兩點最遠
// 距離，請先轉成凸包。若凸包少於兩個點，回傳零。
#define kk (k + 1) % n
ll maxdist(vector<pii>& poly) {
    int k = 1, n = poly.size();
    if (n < 2) return 0;
    if (n == 2) return dd(poly[0], poly[1]);
    ll ret = 0;
    for (int i = 0; i < n; i++) {
        while (abs(crzf(poly[kk], poly[i], poly[ii])) >=
               abs(crzf(poly[k], poly[i], poly[ii])))
            k = kk;
        ret = max(ret, max(dd(poly[i], poly[k]),
                           dd(poly[ii], poly[k])));
    }
    return ret;
}
```

3.9 LCA

```
// b:倍增表 l:路徑 d:算深度跟父節點
int n,m,p,q,de[200001]={},fa[30][200001]={};
vector<int>g[200001];
void d(int w,int l){
    fa[0][w]=l;
    for(auto&k:g[w])if(!de[k]){
        de[k]=de[w]+1;
        d(k,w);
    }
}
void b(int r){
    de[r]=1;
    d(r,r);
    for(int k=1;k<30;k++)for(int i=0;i<n;i++)fa[k][i]=fa[k
        -1][fa[k-1][i]];
}
int l(int p,int q){
    if(de[p]>de[q])swap(p,q);
    int s=de[q]-de[p];
    for(int k=0;k<30;k++)if(s>>k&1)q=fa[k][q];
    if(p==q)return p;
    for(int k=29;k>=0;k--)if(fa[k][p]!=fa[k][q]){p=fa[k][p];q
        =fa[k][q];}
    return fa[0][p];
}
int main(){cin>>n>>m;
    for(int k=1;k<n;k++){cin>>p;g[p-1].push_back(k);}b(0);
    while(m--){cin>>p>>q;cout<<l(p-1,q-1)+1<<'\n';}
}
```

3.10 SPFA

```
typedef pair<int, int> pii;
const int maxn = 1e5 + 5, INF = 0x3f3f3f3f;
vector<pii> G[maxn];
int dis[maxn];
void SPFA(int n, int s) {
    for (int i = 1; i <= n; i++) dis[i] = INF;
    dis[s] = 0; queue<int> q; q.push(s);
    bool inque[maxn] = {};
    while (!q.empty()) {
        int u = q.front(); q.pop();
        inque[u] = false;
        for (pii e : G[u]) {
            int v = e.first, w = e.second;
            if (dis[u] + w < dis[v]) {
                if (!inque[v]) q.push(v), inque[v] = true;
                dis[v] = dis[u] + w;
int main() {
    int n, m, s; cin >> n >> m >> s;
    while (m--) {
        int u, v, w; cin >> u >> v >> w;
        G[u].emplace_back(v, w);
    }
    SPFA(n, s);
}
```

4 Math

4.1 FPow

```
// 問 a ^ p
ll fastpow(ll a, int p) {
    ll ret = 1;
    while (p) {
        if (p & 1) ret *= a;
        a *= a, p >>= 1;
    } return ret;
}
// 問 (a ^ p) mod m
ll fastpow(ll a, ll p, ll m) {
    ll ret = 1;
    while (p) {
        if (p & 1) ret = ret * a % m;
        a = a * a % m, p >>= 1;
    } return ret;
}
```

4.2 Fib

```
1/sqrt(5)*(pow((1+sqrt(5))/2,n)-pow((1-sqrt(5))/2,n))
```

4.3 JosephusProblem

```
// asking last one O(klog(n))
// 編 號 從 1 開 始 ， 結 果 要 加 1
int josephus(int n, int k) {
```

```cpp
    if (k == 1) return n - 1;
    int ans = 0;
    for (int i = 2; i <= n; ) {
        if (ans + k >= i) {
            ans = (ans + k) % i;
            i++;
            continue;
        }
        int step = (i - 1 - ans - 1) / (k - 1); // 向下取整
        if (i + step > n) {
            ans += (n - (i - 1)) * k;
            break;
        }
        i += step;
        ans += step * k;
    }
    return ans;
}


// Josephus Problem
LL J(LL n, const vector<LL>& cs) {
    LL ans = 0, m = cs.size(), sum = 0;
    for (LL v : cs) sum += v;
    if (n == 1 || sum == m) return n - 1;
    for (LL i = 2; i <= n;) {
        LL d = min((n - i + 1) / m, (i - 2 - ans) / (sum - m));
        if (d <= 0) {
            ans = (ans + cs[(n - i) % m]) % i;
            i++;
            continue;
        }
        i += d * m, ans += d * sum;
    }
    return ans;
}
LL slow(LL n, const vector<LL>& cs, LL sum = 0) {
    LL ans = 0;
    for (LL i = 1; i <= n; i++) ans = (ans + cs[(n - i) % cs.size()]) % i;
    return ans;
}

int main() {
    LL n, m;
    while (cin >> n >> m) {
        LL sum = 0;
        vector<LL> cs(m);
        for (auto& v : cs) {
            cin >> v;
            sum += v;
        }
        cout << J(n, cs) + 1 << endl;
        //cout << slow(n, cs, sum) + 1 << endl;
    }
}
```

4.4 SG

```
Anti Nim (取走最後一個石子者敗):
先手必勝 if and only if
1. 「所有」堆的石子數都為 1 且遊戲的 SG 值為 0。
2. 「有些」堆的石子數大於 1 且遊戲的 SG 值不為 0。
Anti-SG (決策集合為空的遊戲者贏):
定義 SG 值為 0 時,遊戲結束,
則先手必勝 if and only if
1. 遊戲中沒有單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數為 0。
2. 遊戲中某個單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數不為 0。
-----------------
Sprague -Grundy :
1. 雙人、回合制
2. 資訊完全公開
3. 無隨機因素
4. 可在有限步內結束
5. 沒有和局
6. 雙方可採取的行動相同

SG(S) 的值為 0:後手 (P) 必勝
不為 0:先手 (N) 必勝

int mex(set S) {
// find the min number >= 0 that not in the S
// e.g. S = {0, 1, 3, 4} mex(S) = 2
}
state = []
int SG(A) {
    if (A not in state) {
        S = sub_states(A)
        if( len(S) > 1 ) state[A] = reduce(operator.xor, [SG(B)for B in S])
        else state[A] = mex(set(SG(B) for B in next_states(A)))
    } return state[A]
}
```

4.5 線篩

```cpp
void sieve(int n) {
    vector<bool> p(n, true);
    vector<int> prime;
    for (int i = 2; i < n; i++) {
        if (p[i]) prime.push_back(i);
        for (auto& k : prime) {
            if (i * k >= n) break;
            p[i * k] = false;
            if (i % k == 0) break;
        }
    }
}
```

4.6 鞋帶

```cpp
vector<pair<int,int>> v(n);
for(auto&n:v) cin >> n.first >> n.second;
int area = 0;
for(int i = 0; i < v.size(); i++){
    area += v[i].first * v[(i+1)%v.size()].second;
    area -= v[i].second * v[(i+1)%v.size()].first;
}
cout << abs(area)/2 << endl;
```

4.7 質因數分解

```cpp
LL func(const LL n, const LL mod, const int c) {
    return (n * n % mod + c + mod) % mod;
}
LL pollorrho(const LL n, const int c) {
    LL a = 1, b = 1;
    a = func(a, n, c) % n;
    b = func(b, n, c) % n;
    b = func(b, n, c) % n;
    while (__gcd(abs(a - b), n) == 1) {
        a = func(a, n, c) % n;
        b = func(b, n, c) % n;
        b = func(b, n, c) % n;
    }
    return __gcd(abs(a - b), n);
}
void prefactor(LL &n, vector<LL> &v, const vector<int> &prime) {
    for (int i = 0; i < 12; ++i) {
        while (n % prime[i] == 0) {
            v.push_back(prime[i]);
            n /= prime[i];
        }
    }
}
void smallfactor(LL n, vector<LL> &v, const vector<int> &isp, const int MAXPRIME, const vector<int> &prime) {
    if (n < MAXPRIME) {
        while (isp[(int)n]) {
            v.push_back(isp[(int)n]);
            n /= isp[(int)n];
        }
        v.push_back(n);
    } else {
        for (int i = 0; i < prime.size() && prime[i] * prime[i] <= n; ++i) {
            while (n % prime[i] == 0) {
                v.push_back(prime[i]);
                n /= prime[i];
            }
        }
        if (n != 1) v.push_back(n);
    }
}
void comfactor(const LL &n, vector<LL> &v, const vector<int> &prime, const vector<int> &isp, const int MAXPRIME) {
    if (n < 1e9) {
        smallfactor(n, v, isp, MAXPRIME, prime);
        return;
    }
    if (Isprime(n)) {
        v.push_back(n);
        return;
    }
    LL d;
    for (int c = 3;; ++c) {
        d = pollorrho(n, c);
        if (d != n) break;
    }
    comfactor(d, v, prime, isp, MAXPRIME);
    comfactor(n / d, v, prime, isp, MAXPRIME);
}
void Factor(const LL &x, vector<LL> &v, const vector<int> &prime, const vector<int> &isp, const int MAXPRIME) {
    LL n = x;
    if (n == 1) {
        puts("Factor 1");
        return;
    }
    prefactor(n, v, prime);
    if (n == 1) return;
    comfactor(n, v, prime, isp, MAXPRIME);
    sort(v.begin(), v.end());
}

void AllFactor(const LL &n, vector<LL> &v, const vector<int> &prime, const vector<int> &isp, const int MAXPRIME) {
    vector<LL> tmp;
    Factor(n, tmp, prime, isp, MAXPRIME);
    v.clear();
    v.push_back(1);
    int len;
    LL now = 1;
    for (int i = 0; i < tmp.size(); ++i) {
        if (i == 0 || tmp[i] != tmp[i - 1]) {
            len = v.size();
            now = 1;
        }
        now *= tmp[i];
        for (int j = 0; j < len; ++j)
            v.push_back(v[j] * now);
    }
}
```

4.8 擴展歐幾里德

```
// 給 a,b ， 解 ax+by=gcd(a,b)
typedef pair<ll, ll> pii;
pii extgcd(ll a, ll b) {
    if (b == 0) return {1, 0};
    ll k = a / b;
    pii p = extgcd(b, a - k * b);
    return {p.second, p.first - k * p.second};
}
```

5 Other

5.1 精度

```
from decimal import*
getcontext().prec = 1000000
n = Decimal(input())
```

5.2 莫隊

```
// 區間眾數
#define ql first.first
#define qr first.second
#define id second
int N, Q, K;
vector<int> A;
vector<pair<pii, int>> qrys;
ll ans[200007];
void init() {
    cin >> N >> Q;
    K = max(1, (int)sqrt(Q)); // Size of the block
    A.clear();
    A.resize(N);
    qrys.clear();
    qrys.resize(Q);

    for (auto &i : A) cin >> i;
    int cnt = 0;
    for (auto &i : qrys) {
        cin >> i.ql >> i.qr;
        --i.ql;
        --i.qr; // Convert to 0-base
        i.id = cnt++;
    }
}
void solve() {
    sort(qrys.begin(), qrys.end(),
        [&](const pair<pii, int> &a, const pair<pii, int> &b) {
            if (a.ql / K == b.ql / K) // Same left block, sort by right block
                return a.qr < b.qr;
            return a.ql / K < b.ql / K; // Otherwise, sort by left block
        });
    int l = 0, r = -1; // Initial answer window
    ll sum = 0;
    for (auto &i : qrys) {
        while (l > i.ql) { l--; sum += A[l]; } // Extend left boundary
        while (r < i.qr) { r++; sum += A[r]; } // Extend right boundary
        while (l < i.ql) { sum -= A[l]; l++; } // Shrink left boundary
        while (r > i.qr) { sum -= A[r]; r--; } // Shrink right boundary
        ans[i.id] = sum;
    }
    for (int i = 0; i < Q; i++) cout << ans[i] << endl;
}
int main() {
    init();
    solve();
    return 0;
}
```

5.3 離散化

```
vector<int> v(1000),b(1000);
for(auto&n:v)cin>>n;
sort(v.begin(),v.end());
auto len = unique(v.begin(),v.end())-v.begin();
v.resize(len);
for(int i = 0; i < v.size(); i++){
    b[i] = lower_bound(v.begin(),v.end(),v[i])-v.begin();
}
```

6 String

6.1 ACAutomaton

```
// Definition of the Aho-Corasick Trie structure
struct _AC {
    _AC* child[26];
    _AC* Fail;
    vector<pair<int, int>> out;
    _AC() {
        Fail = NULL;
        memset(child, 0, sizeof(child));
    }
} *root;
// Function to insert a pattern into the Aho-Corasick Trie
void Insert_AC(string s) {
    int n;
    _AC* p = root;
    for (auto& k : s) {
        n = k - 'a';
        if (!p->child[n]) p->child[n] = new _AC();
        p = p->child[n];
    }
    p->out.push_back({ s.size(), 0 });
}
// Function to construct the Aho-Corasick Trie with failure links
void Construct_AC() {
    queue<_AC*> Q;
    for (int k = 0; k < 26; k++) {
        if (root->child[k]) {
            root->child[k]->Fail = root;
            Q.push(root->child[k]);
        } else {
            root->child[k] = root;
        }
    }
    _AC* p;
    while (!Q.empty()) {
        p = Q.front();
        Q.pop();
        for (int k = 0; k < 26; k++) {
            if (!p->child[k]) {
                p->child[k] = p->Fail->child[k];
            } else {
                p->child[k]->Fail = p->Fail->child[k];
                for (auto& i : p->Fail->child[k]->out)
                    p->child[k]->out.push_back({ i.first, 1 });
                Q.push(p->child[k]);
            }
        }
    }
}
// Function to match patterns in the given text using the Aho-Corasick Trie
void Match_AC(string t) {
    int n;
    _AC* p = root;
    for (int k = 0; k < t.size(); k++) {
        n = t[k] - 'a';
        p = p->child[n];
        _AC* fail = p;
        while (fail != root && fail->out.size()) {
            for (auto& i : fail->out)
                if (!i.second || fail->Fail->out.size())
                    cout << t.substr(k - i.first + 1, i.first) << '\n';
            fail->out.clear();
            fail->Fail->out.clear();
            fail = fail->Fail;
        }
    }
}
// Main function to test the Aho-Corasick Trie
int main() {
    int n, m;
```

```cpp
    string p, t;
    while (cin >> n >> m) {
        root = new _AC();
        while (n--) {
            cin >> p;
            Insert_AC(p);
        }
        Construct_AC();
        cin >> t;
        Match_AC(t);
    }
}
```

6.2 EditDistance

```cpp
ll edd(string& src, string& dst, ll ins, ll del, ll sst) {
    vector<vector<long long>> dp(src.size() + 1, vector<long long>(dst.size() + 1));
    ll dp[src.size() + 1][dst.size() + 1];
    for (int i = 0; i <= src.size(); i++) {
        for (int j = 0; j <= dst.size(); j++) {
            if (i == 0) dp[i][j] = ins * j;
            else if (j == 0) dp[i][j] = del * i;
            else if (src[i - 1] == dst[j - 1]) dp[i][j] = dp[i - 1][j - 1];
            else dp[i][j] = min(dp[i][j - 1] + ins,min(dp[i - 1][j] + del,dp[i - 1][j - 1] + sst));
        }
    }
    return dp[src.size()][dst.size()];
}
```

6.3 KMP

```cpp
// KMP fail function.
int* kmp_fail(string& s) {
    int* f = new int[s.size()];
    int p = f[0] = -1;
    for (int i = 1; i < s.size(); i++) {
        while (p != -1 && s[p + 1] != s[i]) p = f[p];
        if (s[p + 1] == s[i]) p++;
        f[i] = p;
    }
    return f;
}

// Function to count how many times sub appears in str.
int kmp_count(string& str, string& sub) {
    int* fail = kmp_fail(sub);
    int p = -1, ret = 0;
    for (int i = 0; i < str.size(); i++) {
        while (p != -1 && sub[p + 1] != str[i]) p = fail[p];
        if (sub[p + 1] == str[i]) p++;
        if (p == sub.size() - 1) p = fail[p], ret++;
    }
    delete[] fail;
    return ret;
}

//問sub在str第一次出現的開頭index。-1表示找不到 。
int kmp(string& str, string& sub) {
    int* fail = kmp_fail(sub);
    int i = 0, j = 0;
    while (i < str.size() && j < sub.size()) {
        if (sub[j] == str[i]) i++, j++;
        else if (j == 0) i++;
        else j = fail[j - 1] + 1;
    }
    delete[] fail;
    return j == sub.size() ? (i - j) : -1;
}
```

6.4 LPS

```cpp
int lps_length(string s) {
    int N = 2 * s.size() + 1;
    vector<int> dp(N);
    string s2 = "*";
    for (auto& c : s) s2.push_back(c), s2.push_back('*');
    int C = 0, R = 0;
    for (int i = 1; i < N; i++) {
        if (i > R) C = R = i;
        else {
            int mirrorI = 2 * C - i;
            dp[i] = min(dp[mirrorI], R - i);
        }
        int j = dp[i] + 1;
        while ((i - j >= 0) && (i + j < N) && (s2[i - j] == s2[i + j])) j++;
        dp[i] = j - 1;
        if (i + dp[i] > R) C = i, R = i + dp[i];
    }
    return *max_element(dp.begin(), dp.end());
}

string lps(string s) {
    int N = 2 * s.size() + 1;
    vector<int> dp(N);
    string s2 = "*";
    for (auto& c : s) s2.push_back(c), s2.push_back('*');
    int C = 0, R = 0;
    for (int i = 1; i < N; i++) {
        if (i > R) C = R = i;
        else {
            int mirrorI = 2 * C - i;
            dp[i] = min(dp[mirrorI], R - i);
        }
        int j = dp[i] + 1;
        while ((i - j >= 0) && (i + j < N) && (s2[i - j] == s2[i + j])) j++;
        dp[i] = j - 1;
        if (i + dp[i] > R) C = i, R = i + dp[i];
    }
    auto it = max_element(dp.begin(), dp.end());
    int maxLen = *it;
    int index = it - dp.begin();
    return s.substr((index - maxLen) / 2, maxLen);
}
```

6.5 Trie

```cpp
class Trie {
private:
    struct Node {
        int cnt = 0, sum = 0;
        Node* tr[128] = {};  // Array of pointers to child nodes

        ~Node() {
            for (int i = 0; i < 128; i++) {
                if (tr[i]) delete tr[i];
            }
        }
    };

    Node* root;

public:
    // Constructor
    Trie() {
        root = new Node();
    }
    // Destructor
    ~Trie() {
        delete root;
    }
    // Function to insert a string into the Trie
    void insert(char* s) {
        Node* ptr = root;
        for (; *s; s++) {
            if (!ptr->tr[*s]) {
                ptr->tr[*s] = new Node();
            }
            ptr = ptr->tr[*s];
            ptr->sum++;
        }
        ptr->cnt++;
    }
    // Function to count the occurrences of a string in the Trie
    inline int count(char* s) {
        Node* ptr = find(s);
        return ptr ? ptr->cnt : 0;
    }
    // Function to find a node corresponding to a string
    Node* find(char* s) {
        Node* ptr = root;
        for (; *s; s++) {
            if (!ptr->tr[*s]) return nullptr;
            ptr = ptr->tr[*s];
        }
        return ptr;
    }
    // Function to erase a string from the Trie
    bool erase(char* s) {
        if (!count(s)) return false; // If the word doesn't exist, return false

        Node* ptr = root;
        for (char* t = s; *t; t++) {
            Node* tmp = ptr->tr[*t];
            tmp->sum--;
            if (tmp->sum == 0) {
                delete tmp;
                ptr->tr[*t] = nullptr;
                return true;
            }
            ptr = tmp;
        }
        ptr->cnt--; // Decrease the count of the word itself
        return true;
    }
};
```

MCU-NL Codebook

Contents