

# 1 Data Structure

## 1.1 CDQ

```

1 #define fastio ios_base::sync_with_stdio(false); cin.tie(0);
2 #define endl '\n'
3 #define _ << ' ' <<
4 const int INF = 2e9;
5 struct info {
6     int t, p;
7     long long x;
8     // 當 x 界在 -1e9 ~ 1e9，代表這是修改
9     // 當 x >= INF，代表這是詢問
10    // 並且詢問的編號為 (x - INF)
11    // 例：第 2 次詢問，那 x 就設為 INF + 2
12    // 當 x <= -INF，一樣代表詢問，只不過是倒扣
13    // 的詢問
14};
15 int n, q;
16 vector<info> v;
17 vector<long long> ans;
18 void cdq(int l, int r) {
19     if (l == r) return;
20     int mid = (l+r) / 2;
21     cdq(l, mid);
22     cdq(mid+1, r);
23     vector<info> temp;
24     long long add_sum = 0;
25     int pl = l, pr = mid+1;
26     while (pl <= mid && pr <= r) {
27         if (v[pl].p <= v[pr].p) {
28             temp.emplace_back(v[pl]);
29             if (abs(v[pl].x) <= 1e9) {
30                 add_sum += v[pl].x;
31             }
32             pl++;
33         }
34         else {
35             temp.emplace_back(v[pr]);
36             if (v[pr].x >= INF) {
37                 ans[v[pr].x - INF] += add_sum;
38             } else if (v[pr].x <= -INF) {
39                 ans[-(v[pr].x + INF)] -= add_sum;
40             }
41             pr++;
42         }
43     }
44     while (pl <= mid) {
45         temp.emplace_back(v[pl]);
46         pl++;
47     }
48     while (pr <= r) {
49         temp.emplace_back(v[pr]);
50         if (v[pr].x >= INF) ans[v[pr].x - INF] += add_sum;
51         else if (v[pr].x <= -INF) ans[-(v[pr].x + INF)] -= add_sum;
52         pr++;
53     }
54     for (int i = l, j = 0; i <= r; i++, j++) v[i] = temp[j];
55 }
56 int main() {
57     fastio;
58     cin >> n >> q;

```

```

59     v.clear(); ans.clear();
60     int time = 0;
61     vector<long long> arr(n+1);
62     for (int p = 1; p <= n; p++) {
63         long long x; cin >> x;
64         v.emplace_back(info{++time, p, x});
65         arr[p] = x;
66     }
67     int ask_id = 0;
68     for (int i = 0; i < q; i++) {
69         int op; cin >> op;
70         if (op == 1) {
71             int p; long long x;
72             cin >> p >> x;
73             v.emplace_back(info{++time, p, x - arr[p]});
74             arr[p] = x;
75         }
76         else {
77             int l, r; cin >> l >> r;
78             v.emplace_back(info{++time, r, INF + ask_id});
79             if (l > 1) v.emplace_back(info{time, l-1, -INF - ask_id});
80             ask_id++;
81             ans.emplace_back(0);
82         }
83     }
84     cdq(0, (int)v.size()-1);
85     for (int i = 0; i < (int)ans.size(); i++)
86         cout << ans[i] << endl;
87 }
88
89 struct info {
90     int x, y, z, id; // id 是數對在原本陣列的位置
91 };
92 int N;
93 vector<info> v;
94 vector<int> ans;
95 struct BIT {
96     vector<int> bit;
97     int n;
98     BIT(int n) : n(n) bit.resize(n + 1, 0);
99     void upd(int idx, int val) {
100         while (idx <= n) {
101             bit[idx] += val;
102             idx += idx & -idx;
103         }
104     }
105     int qry(int idx) {
106         int sum = 0;
107         while (idx > 0) {
108             sum += bit[idx];
109             idx -= idx & -idx;
110         }
111         return sum;
112     }
113 };
114
115 void cdq(int L, int R, BIT& BITree) {
116     if (L == R) return;
117     int mid = (L + R) / 2;
118     cdq(L, mid, BITree);
119     cdq(mid + 1, R, BITree);
120     vector<info> temp;
121     vector<pair<int, int>> BIT_op; // BIT_op: 紀錄你在BIT的操

```

```

122     int pl = L, pr = mid + 1;
123     while (pl <= mid && pr <= R) {
124         if (v[pl].y < v[pr].y) {
125             temp.emplace_back(v[pl]);
126             BITree.upd(v[pl].z, 1);
127             BIT_op.emplace_back(make_pair(v[pl].z, 1));
128             pl++;
129         } else {
130             temp.emplace_back(v[pr]);
131             ans[v[pr].id] += BITree.qry(v[pr].z - 1);
132             pr++;
133         }
134     }
135     while (pl <= mid) temp.emplace_back(v[pl]), pl++;
136     while (pr <= R) {
137         temp.emplace_back(v[pr]);
138         ans[v[pr].id] += BITree.qry(v[pr].z - 1);
139         pr++;
140     }
141     for (int pt = 0, pv = L; pv <= R; pt++, pv++) v[pv] = temp[pt];
142     for (auto& op : BIT_op) {
143         BITree.upd(op.first, -op.second);
144     }
145 }
146 int main() {
147     fastio;
148     cin >> N;
149     v.resize(N);
150     ans.resize(N, 0);
151     for (int i = 0; i < N; i++) {
152         cin >> v[i].x >> v[i].y >> v[i].z, v[i].id = i;
153     }
154     sort(v.begin(), v.end(), [](const info& a, const info& b) {
155         return a.x < b.x; });
156     BIT BITree(N);
157     cdq(0, N - 1, BITree);
158     for (int i = 0; i < N; i++) cout << ans[i] << endl;
159 }

```

## 1.2 DSU

```

1 class UnionFind {
2 private:
3     vector<int> parent;
4     vector<int> rank;
5 public:
6     UnionFind(int size) : parent(size), rank(size, 1) {
7         for (int i = 0; i < size; ++i) {
8             parent[i] = i;
9         }
10    }
11    int getSize() const return parent.size();
12    int find(int p) {
13        if (p < 0 || p >= parent.size()) throw out_of_range("
14            Out of bound.");
15        while (p != parent[p]) {
16            parent[p] = parent[parent[p]];
17            p = parent[p];
18        }
19        return p;
20    }
21    int findR(int p) {
22        if (p < 0 || p >= parent.size())
23            throw out_of_range("Out of bound.");
24        if (p != parent[p])

```

```

24     parent[p] = findR(parent[p]);
25     return parent[p];
26 }
27 bool isConnected(int p, int q){ return find(p) == find(q)
    ;}
28 void unionElements(int p, int q) {
29     int pRoot = find(p);
30     int qRoot = find(q);
31     if (pRoot == qRoot) return;
32     if (rank[pRoot] < rank[qRoot]) parent[pRoot] = qRoot;
33     else if (rank[qRoot] < rank[pRoot]) parent[qRoot] =
        pRoot;
34     else {
35         parent[pRoot] = qRoot;
36         rank[qRoot]++;
37     }
38 }
39
40 int countConnectedNodes(int p) {
41     int root = find(p);
42     int count = 0;
43     for (int i = 0; i < parent.size(); i++) {
44         if (find(i) == root) count++;
45     }
46     return count;
47 }
48 };
49 int main() {
50     UnionFind uf(10);
51     uf.unionElements(1, 2);
52     uf.unionElements(2, 3);
53     uf.unionElements(4, 5);
54     uf.unionElements(3, 4);
55 }

```

### 1.3 Dag

```

1 void DFS(map<int, queue<int>>& graph, map<int, bool>& visited
    , int place, stack<int>& ans) {
2     if (visited[place]) return;
3     visited[place] = true;
4     while (!graph[place].empty()) {
5         DFS(graph, visited, graph[place].front(), ans);
6         graph[place].pop();
7     }
8     ans.push(place);
9 }

```

### 1.4 Kruskal

```

1 struct Edge {
2     int src, dest, weight;
3     Edge(int s, int d, int w) : src(s), dest(d), weight(w) {}
4 };
5 int findParent(const vector<int>& parent, int i) {
6     return parent[i] == i ? i : findParent(parent, parent[i])
    ;
7 }
8 vector<Edge> kruskalMST(const vector<vector<int>>&
    adjacencyMatrix) {
9     int size = adjacencyMatrix.size();
10    vector<Edge> mst, edges;
11    for(int i = 0; i < size; ++i){
12        for(int j = i + 1; j < size; ++j){
13            if(adjacencyMatrix[i][j] > 0){

```

```

        edges.push_back(Edge(i, j, adjacencyMatrix[i
14            ][j]));
15    }
16    }
17    }
18    sort(edges.begin(), edges.end(), [](const Edge& e1, const
    Edge& e2) {
19        return e1.weight < e2.weight;
20    });
21    vector<int> parent(size);
22    for(int i = 0; i < size; ++i) parent[i] = i;
23    int count = 0;
24    for(const Edge& edge : edges) {
25        if(count == size - 1) break;
26        int srcParent = findParent(parent, edge.src),
            destParent = findParent(parent, edge.dest);
27        if(srcParent != destParent) {
28            mst.push_back(edge);
29            ++count;
30            parent[srcParent] = destParent;
31        }
32    }
33    return mst;
34 }

```

### 1.5 Prim

```

1 struct Edge {
2     int src, dest, weight;
3     Edge(int s, int d, int w) : src(s), dest(d), weight(w) {}
4 };
5 vector<Edge> primMST(const vector<vector<int>>&
    adjacencyMatrix) {
6     int size = adjacencyMatrix.size();
7     vector<bool> visited(size, false);
8     vector<Edge> mst;
9     priority_queue<Edge, vector<Edge>, function<bool(Edge,
    Edge)>> pq(
10         [](const Edge& e1, const Edge& e2) { return e1.weight
            > e2.weight; }
11     );
12     visited[0] = true;
13     for (int i = 1; i < size; ++i) {
14         if (adjacencyMatrix[0][i] > 0) {
15             pq.push(Edge(0, i, adjacencyMatrix[0][i]));
16         }
17     }
18     while (!pq.empty()) {
19         Edge minEdge = pq.top();
20         pq.pop();
21         int u = minEdge.dest;
22         if (visited[u]) continue;
23         visited[u] = true;
24         mst.push_back(minEdge);
25         for (int v = 0; v < size; ++v) {
26             if (adjacencyMatrix[u][v] > 0 && !visited[v]) {
27                 pq.push(Edge(u, v, adjacencyMatrix[u][v]));
28             }
29         }
30     }
31     return mst;
32 }
33
34 int main() {
35     // Example adjacency matrix

```

```

vector<vector<int>> adjacencyMatrix = {
36     {0, 2, 0, 6, 0},
37     {2, 0, 3, 8, 5},
38     {0, 3, 0, 0, 7},
39     {6, 8, 0, 0, 9},
40     {0, 5, 7, 9, 0}
41 };
42
43 vector<Edge> mst = primMST(adjacencyMatrix);
44 cout << "Minimum Spanning Tree Edges:\n";
45 for (const auto& edge : mst) {
46     cout << "Src: " << edge.src << ", Dest: " << edge.
        dest << ", Weight: " << edge.weight << "\n";
47 }
48
49 return 0;
50 }

```

### 1.6 SegmentTreeLazy

```

1 #define Ls(x) x << 1
2 #define Rs(x) x << 1 | 1
3 int N = 1e5+5;
4 vector<int> segTree(4*N), lazy(4*N) , arr(N);
5 void build(int l, int r, int idx = 1){
6     if(l == r){
7         segTree[idx] = arr[l];
8         return;
9     }
10    int mid = (l+r) >> 1;
11    build(l, mid, Ls(idx));
12    build(mid+1, r, Rs(idx));
13    segTree[idx] = segTree[Ls(idx)]+ segTree[Rs(idx)];
14 }
15 void pushDown(int l, int r, int idx){
16     if(lazy[idx] == 0) return;
17     segTree[idx] += (r-l+1)*lazy[idx];
18     if(l != r) lazy[Ls(idx)] += lazy[idx], lazy[Rs(idx)] +=
        lazy[idx];
19     lazy[idx] = 0;
20 }
21 void update(int l, int r, int ql, int qr, int val, int idx =
    1){
22     pushDown(l, r, idx);
23     if(l > qr || r < ql) return;
24     if(l >= ql && r <= qr){
25         segTree[idx] += (r-l+1)*val;
26         if(l != r) lazy[Ls(idx)] += val, lazy[Rs(idx)] += val
            ;
27         return;
28     }
29     int mid = (l+r) >> 1;
30     update(l, mid, ql, qr, val, Ls(idx));
31     update(mid+1, r, ql, qr, val, Rs(idx));
32     segTree[idx] = segTree[Ls(idx)]+ segTree[Rs(idx)];
33 }
34
35 int query(int l, int r, int ql, int qr, int idx = 1){
36     pushDown(l, r, idx);
37     if(l > qr || r < ql) return 0;
38     if(l >= ql && r <= qr) return segTree[idx];
39     int mid = (l+r) >> 1, sum = 0;
40     if(ql <= mid) sum += query(l, mid, ql, qr, Ls(idx));
41     if(qr > mid) sum += query(mid+1, r, ql, qr, Rs(idx));
42     return sum;
43 }

```

```

44 int32_t main(){
45     int n, t;
46     cin >> n >> t;
47     for(int i = 1; i <= n; i++) cin >> arr[i];
48     build(1, n, 1);
49     for(int i = 0; i < t; i++){
50         int j, l, r, v;
51         cin >> j;
52         if(j == 1){
53             cin >> l >> r >> v;
54             update(1, n, l, r, v);
55         } else {
56             cin >> r;
57             cout << query(r, 1, n) << endl;
58         }
59     }
60 }
61 // prefix sum
62 int N = 2e5 + 5;
63 vector<ll> segTree(4 * N, -2e18), lazy(4 * N), arr(N), tmp(N)
64 ;
65 void build(int l, int r, int idx = 1){
66     if(l == r){
67         segTree[idx] = arr[l];
68         return;
69     }
70     int mid = (l + r) >> 1;
71     build(l, mid, Ls(idx));
72     build(mid + 1, r, Rs(idx));
73     segTree[idx] = max(segTree[Ls(idx)] + segTree[Rs(idx)]);
74 }
75 void pushDown(int l, int r, int idx, int x = 0){
76     if(lazy[idx] == 0) return;
77     segTree[idx] += lazy[idx];
78     int mid = (l + r) / 2;
79     if(l != r) lazy[Ls(idx)] += lazy[idx], lazy[Rs(idx)] +=
80         lazy[idx];
81     lazy[idx] = 0;
82 }
83 void update(int l, int r, int ql, int qr, int val, int idx =
84 1){
85     pushDown(l, r, idx);
86     if(l > qr || r < ql) return;
87     if(l >= ql && r <= qr){
88         segTree[idx] += val;
89         pushDown(l, r, idx);
90         return;
91     }
92     int mid = (l + r) >> 1;
93     update(l, mid, ql, qr, val, Ls(idx));
94     update(mid + 1, r, ql, qr, val, Rs(idx));
95     segTree[idx] = max(segTree[Ls(idx)], segTree[Rs(idx)]);
96 }
97 ll query(int l, int r, int ql, int qr, int idx = 1){
98     pushDown(l, r, idx);
99     if(l > qr || r < ql) return -2e18;
100     if(l >= ql && r <= qr) return segTree[idx];
101     int mid = (l + r) >> 1, ans = -2e18;
102     ans = max(ans, query(l, mid, ql, qr, Ls(idx)));
103     ans = max(ans, query(mid + 1, r, ql, qr, Rs(idx)));
104     segTree[idx] = max(segTree[Ls(idx)], segTree[Rs(idx)]);
105     return ans;
106 }
107 int main(){
108     IO;
109     int n, q;

```

```

107     cin >> n >> q;
108     for(int i = 1; i <= n; i++) cin >> tmp[i];
109     arr = tmp;
110     for(int i = 2; i <= n; i++) arr[i] += arr[i - 1];
111     build(1, n, 1);
112     while(q--){
113         int j, k, u;
114         cin >> j >> k >> u;
115         if(j == 1){
116             update(1, n, k, n, u - tmp[k]), tmp[k] = u;
117         } else {
118             cout << max(0ll, query(1, n, k, u) - (k!=1?query
119                 (1, n, k - 1, k - 1):0)) << endl;
120         }
121     }

```

## 1.7 SegmentTree

```

1 ll v[1e5], seg[4e5+7];
2 void StructSEG(int l, int r, ll node){
3     if(l==r) seg[node] = v[r];
4     else{
5         int mid = (l+r) >> 1, tmp = node << 1;
6         StructSEG(l,mid,tmp);
7         StructSEG(mid+1,r,tmp+1);
8         seg[node] = seg[tmp] + seg[tmp+1];
9     }
10 }
11 int query(int tL, int tR, int nL, int nR, int node){
12     int mid = (nL+nR)>>1, ans = 0;
13     if(tL <= nL && nR <= tR) return seg[node];
14     if(tL <= mid) ans += query(tL,tR,nL, mid,(node<<1));
15     if(tR > mid) ans += query(tL,tR,mid+1,nR,(node<<1)+1);
16     return ans;
17 }
18 void updateNode(int idx, int val, int l, int r, int node)
19 {
20     if(l == r) seg[node] = val, v[idx] = val;
21     else{
22         int m = (l + r) >> 1;
23         int leftNode = (node << 1);
24         int rightNode = (node << 1) + 1;
25         if(idx <= m && idx >= 1) updateNode(idx, val, l, m,
26             leftNode);
27         else updateNode(idx, val, m+1,r, rightNode);
28         seg[node] = seg[leftNode] + seg[rightNode];
29     }

```

## 2 DP

### 2.1 BellmanFord

```

1 typedef pair<int, int> pii;
2 const int maxn = 1e5 + 5;
3 const int INF = 0x3f3f3f3f;
4 vector<pii> G[maxn];
5 int dis[maxn];
6 bool BellmanFord(int n, int s) {
7     for (int i = 1; i <= n; i++) dis[i] = INF;
8     dis[s] = 0;
9     bool relax;
10    for (int r = 1; r <= n; r++) {

```

```

11        relax = false;
12        for (int i = 1; i <= n; i++) {
13            for (pii e : G[i]) {
14                if (dis[i] + e.second < dis[e.first]) {
15                    dis[e.first] = dis[i] + e.second;
16                    relax = true;
17                }
18            }
19        }
20        if (!relax) break; // Optimization: If no relaxation
21        // happened, break early
22    }
23    return relax; // If relax is true after n iterations,
24    // there is a negative cycle
25 }
26 int main() {
27     int n, m, s;
28     cin >> n >> m >> s;
29     while (m--) {
30         int u, v, w;
31         cin >> u >> v >> w;
32         G[u].emplace_back(v, w);
33     }
34     if (BellmanFord(n, s)) {
35         cout << "negative cycle\n";
36     } else {
37         for (int i = 1; i <= n; i++) {
38             if (dis[i] == INF) {
39                 cout << "INF\n"; // Print INF if there's no
40                 // path
41             } else {
42                 cout << dis[i] << " \n"[i == n];
43             }
44         }
45     }

```

### 2.2 EdmondsKarp

```

1 int main() {
2     int n, m, b, e, s, t, w, q = 1;
3     while (cin >> n, n) {
4         int ans = 0, go = 1, Min;
5         vector<vector<int>> a(n+1, vector<int>(n+1, 0));
6         vector<int> last(n+1, 0), can(n+1, 0);
7         cin >> b >> e >> m;
8         while (m--) {
9             cin >> s >> t >> w;
10            a[s][t] = a[t][s] += w;
11        }
12        while (go) {
13            go = 0;
14            queue<int> Q;
15            Q.push(b);
16            Min = 99999999;
17            fill(can.begin(), can.end(), 0);
18            while (Q.size()) {
19                int top = Q.front();
20                Q.pop();
21                can[top] = 1;
22                for (int k = 1; k <= n; k++) {
23                    if (a[top][k] && !can[k]) {
24                        last[k] = top;
25                        if (k == e) {
26                            go = 1;

```

```

27         break;
28     }
29     Q.push(k);
30 }
31 }
32 if (go) {
33     int S = s = e;
34     while (last[s]) {
35         Min = min(Min, a[last[s]][s]);
36         s = last[s];
37     }
38     while (last[S]) {
39         a[last[S]][S] -= Min;
40         a[S][last[S]] += Min;
41         S = last[S];
42     }
43     ans += Min;
44     fill(last.begin(), last.end(), 0);
45     fill(can.begin(), can.end(), 0);
46     break;
47 }
48 }
49 printf("Network %d\nThe bandwidth is %d.\n", q++,
50 ans);
51 }
52 }

```

## 2.3 LCS

1 若一樣：左上 +1  
2 else max(左, 上)

## 2.4 LCS2 LIS

```

1 int LCS2LIS(string &a, string &b) {
2     vector<int> List;
3     for (int i = 0; i < b.size(); i++) {
4         for (int j = 0; j < a.size(); j++) if (b[i] == a[j])
5             List.push_back(j);
6     }
7     if (List.size() == 0) return 0;
8     vector<int> dp;
9     for (int i = 0; i < List.size(); i++) {
10        if (dp.empty() || List[i] > dp.back()) dp.push_back(
11            List[i]);
12        else *lower_bound(dp.begin(), dp.end(), List[i]) =
13            List[i];
14    }
15    return dp.size();
16 }

```

## 2.5 LCS3D

```

1 // space O(m*n*o) version
2 int lcsOf3(string X, string Y, string Z, int m, int n, int o)
3 {
4     int L[m+1][n+1][o+1];
5     for (int i = 0; i <= m; i++) {
6         for (int j = 0; j <= n; j++) {
7             for (int k = 0; k <= o; k++) {
8                 if (i == 0 || j == 0 || k == 0) {
9                     L[i][j][k] = 0;

```

```

9                 } else if (X[i-1] == Y[j-1] && X[i-1] == Z[k
10                    -1]) {
11                     L[i][j][k] = L[i-1][j-1][k-1] + 1;
12                 } else {
13                     L[i][j][k] = max({L[i-1][j][k], L[i][j
14                        -1][k], L[i][j][k-1]});
15                 }
16             }
17         }
18     }
19     return L[m][n][o];
20 }
21 // space O(2*n*o) version
22 int lcsOf3(string X, string Y, string Z, int m, int n, int o)
23 {
24     vector<vector<vector<int>>> L(2, vector<vector<int>>(n+1,
25        vector<int>(o+1, 0)));
26     for (int i = 0; i <= m; i++) {
27         for (int j = 0; j <= n; j++) {
28             for (int k = 0; k <= o; k++) {
29                 if (i == 0 || j == 0 || k == 0) {
30                     L[i % 2][j][k] = 0;
31                 } else if (X[i-1] == Y[j-1] && X[i-1] == Z[k
32                    -1]) {
33                     L[i % 2][j][k] = L[(i-1) % 2][j-1][k-1] +
34                         1;
35                 } else {
36                     L[i % 2][j][k] = max({L[(i-1) % 2][j][k],
37                        L[i % 2][j-1][k], L[i % 2][j][k
38                           -1]});
39                 }
40             }
41         }
42     }
43     return L[m % 2][n][o];
44 }

```

## 2.6 LIS

```

1 void LIS(vector<int> &arr) {
2     vector<int> lis;
3     for (int i = 0; i < arr.size(); i++) {
4         auto it = lower_bound(lis.begin(), lis.end(), arr[i])
5             ;
6         if (it == lis.end()) lis.push_back(arr[i]);
7         else *it = arr[i];
8     }
9     cout << lis.size() << endl;
10 }

```

## 2.7 MaxFlow

```

1 struct edge {
2     int from, to, cap, flow;
3     edge(int u, int v, int c, int f) : from(u), to(v), cap(c)
4         , flow(f) {}
5 };
6 struct Dinic {
7     vector<edge> edges;
8     vector<vector<int>> adj;
9     vector<int> level, ptr;
10    int n, m = 0, s, t;
11    Dinic(int n, int s, int t) : n(n), s(s), t(t) {
12        adj.resize(n);

```

```

13        level.resize(n);
14        ptr.resize(n);
15    }
16    void addEdge(int u, int v, int c) {
17        edges.emplace_back(u, v, c, 0);
18        edges.emplace_back(v, u, 0, 0);
19        adj[u].push_back(m);
20        adj[v].push_back(m + 1);
21        m += 2;
22    }
23    bool bfs() {
24        fill(level.begin(), level.end(), -1);
25        queue<int> q;
26        q.push(s);
27        level[s] = 0;
28        while (!q.empty()) {
29            int u = q.front();
30            q.pop();
31            for (int id : adj[u]) {
32                if (edges[id].cap - edges[id].flow < 1)
33                    continue;
34                int v = edges[id].to;
35                if (level[v] != -1) continue;
36                level[v] = level[u] + 1;
37                q.push(v);
38            }
39        }
40        return level[t] != -1;
41    }
42    int dfs(int u, int pushed) {
43        if (pushed == 0) return 0;
44        if (u == t) return pushed;
45        for (int& cid = ptr[u]; cid < (int)adj[u].size(); cid
46            ++){
47            int id = adj[u][cid];
48            int v = edges[id].to;
49            if (level[u] + 1 != level[v] || edges[id].cap -
50                edges[id].flow < 1) continue;
51            int tr = dfs(v, min(pushed, edges[id].cap - edges
52                [id].flow));
53            if (tr == 0) continue;
54            edges[id].flow += tr;
55            edges[id ^ 1].flow -= tr;
56            return tr;
57        }
58        return 0;
59    }
60    int flow() {
61        int f = 0;
62        while (true) {
63            if (!bfs()) break;
64            fill(ptr.begin(), ptr.end(), 0);
65            while (int pushed = dfs(s, INT_MAX)) f += pushed;
66        }
67        return f;
68    }
69 }
70 int main() {
71     int n, m, s, t;
72     cin >> n >> m >> s >> t;
73     Dinic dinic(n, s, t);
74     for (int i = 0; i < m; i++) {
75         int u, v, c;
76         cin >> u >> v >> c;
77         dinic.addEdge(u, v, c);
78     }

```

```
74 cout << dinic.flow() << endl;
75 }
```

## 2.8 SPFA

```
1 typedef pair<int, int> pii;
2 const int maxn = 1e5 + 5;
3 const int INF = 0x3f3f3f3f;
4 vector<pii> G[maxn];
5 int dis[maxn];
6
7 void SPFA(int n, int s) {
8     for (int i = 1; i <= n; i++) dis[i] = INF;
9     dis[s] = 0;
10    queue<int> q;
11    q.push(s);
12    bool inque[maxn] = {};
13    inque[s] = true;
14    while (!q.empty()) {
15        int u = q.front();
16        q.pop();
17        inque[u] = false;
18        for (pii e : G[u]) {
19            int v = e.first, w = e.second;
20            if (dis[u] + w < dis[v]) {
21                dis[v] = dis[u] + w;
22                if (!inque[v]) {
23                    q.push(v);
24                    inque[v] = true;
25                }
26            }
27        }
28    }
29 }
30 int main() {
31     int n, m, s;
32     cin >> n >> m >> s;
33     while (m--) {
34         int u, v, w;
35         cin >> u >> v >> w;
36         G[u].emplace_back(v, w);
37     }
38     SPFA(n, s);
39     for (int i = 1; i <= n; i++) {
40         if (dis[i] == INF) cout << "INF ";
41         else cout << dis[i] << " ";
42     }
43 }
```

## 2.9 DP

```
1 #include <iostream>
2 #define N 205
3 #define MAX 0x03ffffff
4 using namespace std;
5
6 int Metrix[N][N], dist[N], vist[N], path[N];
7 struct edge { int s, t, cost; } E[N];
8
9 void OutPath(int s, int t, int NV) {
10     for (int i = 1; i <= NV; i++) cout << path[i] << " ";
11     cout << endl;
12     int u = s, v = t;
13     while (v != s) {
14         cout << v << "-->";
```

```
15         v = path[v];
16     }
17     cout << u << endl;
18 }
19
20 int Dijkstra(int s, int t, int NV) {
21     for (int i = 1; i <= NV; i++) {
22         dist[i] = MAX;
23         vist[i] = 0;
24         path[i] = i;
25     }
26     dist[s] = 0;
27     for (int i = 1; i <= NV; i++) {
28         int min_value = MAX, u = -1;
29         for (int j = 1; j <= NV; j++)
30             if (!vist[j] && dist[j] < min_value)
31                 min_value = dist[j], u = j;
32         if (u == -1) break;
33         vist[u] = 1;
34         for (int j = 1; j <= NV; j++)
35             if (!vist[j] && dist[u] + Metrix[u][j] < dist[j])
36                 dist[j] = dist[u] + Metrix[u][j], path[j] = u;
37     }
38     return dist[t] == MAX ? -1 : dist[t];
39 }
40
41 void SP2th(int s, int t, int NV) {
42     int flag = Dijkstra(s, t, NV);
43     if (flag == -1) {
44         cout << "不可达" << endl;
45         return;
46     }
47     int u = s, v = t, arcNum = 0;
48     while (v != u) {
49         cout << v << "-->";
50         E[arcNum++] = {v, path[v], Metrix[v][path[v]]};
51         v = path[v];
52     }
53     cout << u << endl << ":" << dist[t] << endl;
54     int min_SP = MAX;
55     for (int i = 0; i < arcNum; i++) {
56         u = E[i].s; v = E[i].t;
57         Metrix[u][v] = Metrix[v][u] = MAX;
58         flag = Dijkstra(s, t, NV);
59         if (flag != -1) {
60             OutPath(s, t, NV);
61             cout << ":" << dist[t] << endl;
62             if (min_SP > dist[t]) min_SP = dist[t];
63         }
64         Metrix[u][v] = Metrix[v][u] = E[i].cost;
65     }
66     cout << "次短路：" << min_SP << endl;
67 }
68
69 int main() {
70     int m, n, s, t, c;
71     while (cin >> n >> m) {
72         for (int i = 1; i <= n; i++)
73             for (int j = 1; j <= n; j++)
74                 Metrix[i][j] = (i == j) ? 0 : MAX;
75         for (int i = 0; i < m; i++) {
76             cin >> s >> t >> c;
77             Metrix[s][t] = Metrix[t][s] = min(Metrix[t][s], c);
78         }
79     }
```

```
78     }
79     cin >> s >> t;
80     SP2th(s, t, n); //求s->t的次短路
81 }
82 }
```

## 3 Geometry

### 3.1 ClosestPair

```
1 typedef pair<ll, ll> pii;
2 #define x first
3 #define y second
4 ll dd(const pii& a, const pii& b) {
5     ll dx = a.x - b.x, dy = a.y - b.y;
6     return dx * dx + dy * dy;
7 }
8 const ll inf = 1e18;
9 ll dac(vector<pii>& p, int l, int r) {
10    if (l >= r) return inf;
11    int m = (l + r) / 2;
12    ll d = min(dac(p, l, m), dac(p, m + 1, r));
13    vector<pii> t;
14    for (int i = m; i >= l && p[m].x - p[i].x < d; i--)
15        t.push_back(p[i]);
16    for (int i = m + 1; i <= r && p[i].x - p[m].x < d; i++)
17        t.push_back(p[i]);
18    sort(t.begin(), t.end(),
19        [](pii& a, pii& b) { return a.y < b.y; });
20    int n = t.size();
21    for (int i = 0; i < n - 1; i++)
22        for (int j = 1; j < 4 && i + j < n; j++)
23            // 這裡可以知道是哪兩點是最小點對
24            d = min(d, dd(t[i], t[i + j]));
25    return d;
26 }
27 // 給一堆點，求最近點對的距離「的平方」。
28 ll closest_pair(vector<pii>& pp) {
29     sort(pp.begin(), pp.end());
30     return dac(pp, 0, pp.size() - 1);
31 }
```

### 3.2 ConvexHull

```
1 void Dhull(vector<pair<double, double>> points, vector<pair<
2     double, double>> &hull, int e) {
3     if (e) hull.push_back(points[0]); hull.push_back(points[1]);
4     for (int i = 2; i < points.size(); i++) {
5         while (hull.size() >= 2) {
6             pair<double, double> p1 = hull[hull.size() - 2], p2 = hull
7                 [hull.size() - 1], p3 = points[i];
8             double x1 = p2.first - p1.first, y1 = p2.second - p1.
9                 second;
10            double x2 = p3.first - p2.first, y2 = p3.second - p2.
11                second;
12            if (x1 * y2 - x2 * y1 <= 0) break;
13            hull.pop_back();
14        }
15        hull.push_back(points[i]);
16    }
17 }
18 int main() {
19     string s, ss;
20     while (getline(cin, s)) {
```

```

17 double n,x,y,t=0;char c;
18 vector<pair<double,double>>points,hull;
19 istream sin(s);
20 while(sin>>c>>x>>y>>c){
21     points.push_back({x,y});
22 }
23 sort(points.begin(),points.end(),[(auto a,auto b){if
24     (a.second==b.second)return a.first<b.first;
25     return a.second<b.second;});
26 Dhull(points,hull,1);
27 reverse(points.begin(),points.end());
28 Dhull(points,hull,0);
29 reverse(hull.begin(),hull.end());
30 for(auto&k:hull){
31     if(t++)cout<<' ';
32     cout<<'('<<k.first<<','<<k.second<<')';
33 }

```

### 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: 如果恰有一點の出度等於入度+1、另有一點の入度等於出度+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

```

1 const int maxn = 1000; // Example maximum number of nodes,
2   adjust as needed
3 int n, m, vis[maxn], level[maxn], pr[maxn], pr2[maxn];
4 vector<int> edge[maxn]; // Adjacency list for the left
5   partition
6
7 bool dfs(int u) {
8     vis[u] = true;
9     for (int v : edge[u]) {
10        int pv = pr2[v];
11        if (pv == -1 || (!vis[pv] && level[u] < level[pv] &&
12            dfs(pv))) {
13            pr[u] = v;
14            pr2[v] = u;
15            return true;
16        }
17    }
18    return false;
19 }
20
21 int hopcroftKarp() {
22     memset(pr, -1, sizeof(pr));
23     memset(pr2, -1, sizeof(pr2));
24     int match = 0;
25     while (true) {
26         queue<int> Q;
27         for (int i = 1; i <= n; ++i) {
28             if (pr[i] == -1) {
29                 level[i] = 0;
30                 Q.push(i);
31             } else level[i] = -1;
32         }
33         while (!Q.empty()) {

```

```

34         int u = Q.front(); Q.pop();
35         for (int v : edge[u]) {
36             int pv = pr2[v];
37             if (pv != -1 && level[pv] < 0) {
38                 level[pv] = level[u] + 1;
39                 Q.push(pv);
40             }
41         }
42         memset(vis, 0, sizeof(vis));
43         int d = 0;
44         for (int i = 1; i <= n; ++i) if (pr[i] == -1 && dfs(i))
45             ++d;
46         if (d == 0) return match;
47         match += d;
48     }
49 }
50
51 int main() {
52     cin >> n >> m; // n: number of nodes in the left
53     partition, m: number of nodes in the right partition
54     int edges;
55     cin >> edges;
56     for (int i = 0; i < edges; i++) {
57         int u, v;
58         cin >> u >> v;
59         edge[u].push_back(v + n); // Shift right partition
60         // indices by n
61         edge[v + n].push_back(u); // Add reverse edge for
62         // symmetry
63     }
64     cout << "Maximum number of matches: " << hopcroftKarp()
65     << endl;
66 }

```

### 3.5 Hungarian

```

1 const int INF = 2e9;
2 const int N = 1000; // Example maximum number of nodes,
3   adjust as needed
4 int vis[N], m[N];
5 vector<int> g[N]; // Adjacency list for the bipartite graph
6
7 int dfs(int s) {
8     for (int x : g[s]) {
9         if (vis[x]) continue;
10        vis[x] = 1;
11        if (m[x] == -1 || dfs(m[x])) {
12            m[x] = s;
13            m[s] = x;
14            return 1;
15        }
16    }
17    return 0;
18 }
19
20 int hungarian(int p) { // p: number of women
21     memset(m, -1, sizeof(m));
22     int c = 0;
23     for (int i = 0; i < p; ++i) {
24         if (m[i] == -1) {
25             memset(vis, 0, sizeof(vis));
26             c += dfs(i);
27         }
28     }
29     return c; // Number of successful matches

```

```

28 }
29 int main() {
30     int p, q, edges;
31     cin >> p >> q >> edges; // p: number of women, q: number
32     // of men, edges: number of edges
33     for (int i = 0; i < edges; i++) {
34         int u, v;
35         cin >> u >> v;
36         g[u].push_back(v + p); // Shift men's indices by p
37         g[v + p].push_back(u);
38     }
39     cout << "Maximum number of matches: " << hungarian(p) <<
40     endl;

```

### 3.6 MinCircle

```

1 using PT = point<T>;
2 using CPT = const PT;
3 PT circumcenter(CPT &a, CPT &b, CPT &c) {
4     PT u = b-a, v = c-a;
5     T c1 = u.abs2()/2, c2 = v.abs2()/2;
6     T d = u.cross(v);
7     return PT(a.x+(v.y*c1-u.y*c2)/d, a.y+(u.x*c2-v.x*c1)/d);
8 }
9 void solve(PT p[], int n, PT &c, T &r2){
10    random_shuffle(p,p+n);
11    c = p[0]; r2 = 0; // c, r2 = 圓心, 半徑平方
12    for(int i=1; i<n; i++){
13        if( (p[i]-c).abs2() > r2) {
14            c=p[i]; r2=0;
15            for(int j=0; j<i; j++){
16                if( (p[j]-c).abs2() > r2) {
17                    c.x = (p[i].x+p[j].x)/2;
18                    c.y = (p[i].y+p[j].y)/2;
19                    r2 = (p[j]-c).abs2();
20                    for(int k=0; k<j; k++){
21                        if( (p[k]-c).abs2() > r2) {
22                            c = circumcenter(p[i], p[j], p[k]);
23                            r2 = (p[i]-c).abs2();
24                        }
25                    }
26                }
27            }

```

### 3.7 Angle

```

1 // 計算每個內角角度
2 // 有序的點集合, 計算每個內角角度
3 // 逆時針方向, 從第一個點開始
4 // 0 <= angle <= 2*PI
5 vector<double> angle(const vector<pair<int, int>>& v) {
6     vector<double> ret;
7     for (int i = 0; i < v.size(); i++) {
8         int x1 = v[(i - 1 + v.size()) % v.size()].first - v[i].first;
9         int y1 = v[(i - 1 + v.size()) % v.size()].second - v[i].second;
10        int x2 = v[(i + 1) % v.size()].first - v[i].first;
11        int y2 = v[(i + 1) % v.size()].second - v[i].second;
12        double a = atan2(y1, x1) - atan2(y2, x2);
13        if (a < 0) a += 2 * M_PI;
14        ret.push_back(a);
15        // ret = ret*180/M_PI // convert to degree

```

```

16     }
17     return ret;
18 }
19 // 計算三個點的內角角度
20 // 0 <= angle <= 2*PI
21 double angle(const pair<int, int>& a, const pair<int, int>& b
22             , const pair<int, int>& c) {
23     int x1 = a.first - b.first;
24     int y1 = a.second - b.second;
25     int x2 = c.first - b.first;
26     int y2 = c.second - b.second;
27     double ret = atan2(y1, x1) - atan2(y2, x2);
28     if (ret < 0) ret += 2 * M_PI;
29     return ret;
30     // ret = ret*180/M_PI // convert to degree
31 }
32 // 求兩線交點 (兩線必相交)
33 // 0 <= angle <= 2*PI
34 pair<double, double> intersection(const pair<double, double>&
35 p1, const pair<double, double>& p2, const pair<double,
36 double>& q1, const pair<double, double>& q2) {
37     double a1 = p2.second - p1.second;
38     double b1 = p1.first - p2.first;
39     double c1 = a1 * p1.first + b1 * p1.second;
40     double a2 = q2.second - q1.second;
41     double b2 = q1.first - q2.first;
42     double c2 = a2 * q1.first + b2 * q1.second;
43     double det = a1 * b2 - a2 * b1;
44     return make_pair((b2 * c1 - b1 * c2) / det, (a1 * c2 - a2

```

### 3.8 旋轉卡尺

```

1 typedef pair<ll, ll> pii;
2 #define x first
3 #define y second
4 #define ii (i + 1) % n // 打字加速!
5 inline pii operator-(const pii& a, const pii& b) {
6     return {a.x - b.x, a.y - b.y};
7 } // const 不可省略
8 inline ll operator*(const pii& a, const pii& b) {
9     return a.x * b.y - a.y * b.x;
10 }
11 inline ll crzf(const pii& o, const pii& a, const pii& b) {
12     return (a - o) * (b - o)
13 }
14 inline ll dd(const pii& a, const pii& b) {
15     ll dx = a.x - b.x, dy = a.y - b.y;
16     return dx * dx + dy * dy;
17 }
18 // 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
19 // 重複點。
20 #define jud \
21 crzf(ret[ret.size() - 2], ret.back(), pp[i]) <= 0
22 vector<pii> makepoly(vector<pii>& pp) {
23     int n = pp.size();
24     sort(pp.begin(), pp.end());
25     pp.erase(unique(pp.begin(), pp.end()), pp.end());
26     vector<pii> ret;
27     for (int i = 0; i < n; i++) {
28         while (ret.size() >= 2 && jud) ret.pop_back();

```

```

28     ret.push_back(pp[i]);
29 }
30 for (int i = n - 2, t = ret.size() + 1; i >= 0; i--) {
31     while (ret.size() >= t && jud) ret.pop_back();
32     ret.push_back(pp[i]);
33 }
34 if (n >= 2) ret.pop_back();
35 return ret;
36 }
37 // (shoelace formula)
38 // 給凸包, 問其面積「的兩倍」。若凸包少於三個點, 回傳零。
39 ll area(vector<pii>& poly) {
40     int n = poly.size();
41     ll ret = 0;
42     for (int i = 0; i < n; i++)
43         ret += (poly[i].x * poly[i+1].y);
44     for (int i = 0; i < n; i++)
45         ret -= (poly[i].y * poly[i+1].x);
46     return ret;
47 }
48 // 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
49 // 兩點最遠
50 // 距離, 請先轉成凸包。若凸包少於兩個點, 回傳零。
51 #define kk (k + 1) % n
52 ll maxdist(vector<pii>& poly) {
53     int k = 1, n = poly.size();
54     if (n < 2) return 0;
55     if (n == 2) return dd(poly[0], poly[1]);
56     ll ret = 0;
57     for (int i = 0; i < n; i++) {
58         while (abs(crzf(poly[kk], poly[i], poly[i+1])) >=
59                abs(crzf(poly[k], poly[i], poly[i+1])))
60             k = kk;
61         ret = max(ret, max(dd(poly[i], poly[k]),
62                          dd(poly[i+1], poly[k])));
63     }
64     return ret;
65 }

```

### 3.9 LCA

```

1 // b:倍增表 l:路徑 d:算深度跟父節點
2 int n, m, p, q, de[200001]={}, fa[30][200001]={};
3 vector<int>g[200001];
4 void d(int w, int l){
5     fa[0][w]=1;
6     for(auto&k:g[w])if(!de[k]){
7         de[k]=de[w]+1;
8         d(k, w);
9     }
10 }
11 void b(int r){
12     de[r]=1;
13     d(r, r);
14     for(int k=1;k<30;k++)for(int i=0;i<n;i++)fa[k][i]=fa[k
15     -1][fa[k-1][i]];
16 }
17 int l(int p, int q){
18     if(de[p]>de[q])swap(p, q);
19     int s=de[q]-de[p];
20     for(int k=0;k<30;k++)if(s>>k&1)q=fa[k][q];
21     if(p==q)return p;
22     for(int k=29;k>=0;k--)if(fa[k][p]!=fa[k][q]){p=fa[k][p];q
23     =fa[k][q];}
24     return fa[0][p];

```

```

23 }
24 int main(){cin>>n>>m;
25     for(int k=1;k<n;k++){cin>>p;g[p-1].push_back(k);}b(0);
26     while(m--){cin>>p>>q;cout<<l(p-1,q-1)+1<<'\n';}
27 }

```

### 3.10 SPFA

```

1 typedef pair<int, int> pii;
2 const int maxn = 1e5 + 5, INF = 0x3f3f3f3f;
3 vector<pii> G[maxn];
4 int dis[maxn];
5 void SPFA(int n, int s) {
6     for (int i = 1; i <= n; i++) dis[i] = INF;
7     dis[s] = 0; queue<int> q; q.push(s);
8     bool inque[maxn] = {};
9     while (!q.empty()) {
10         int u = q.front(); q.pop();
11         inque[u] = false;
12         for (pii e : G[u]) {
13             int v = e.first, w = e.second;
14             if (dis[u] + w < dis[v]) {
15                 if (!inque[v]) q.push(v), inque[v] = true;
16                 dis[v] = dis[u] + w;
17             }
18         }
19     }
20     SPFA(n, s);
21 }

```

## 4 Math

### 4.1 FPow

```

1 // 問 a ^ p
2 ll fastpow(ll a, int p) {
3     ll ret = 1;
4     while (p) {
5         if (p & 1) ret *= a;
6         a *= a, p >>= 1;
7     } return ret;
8 }
9 // 問 (a ^ p) mod m
10 ll fastpow(ll a, ll p, ll m) {
11     ll ret = 1;
12     while (p) {
13         if (p & 1) ret = ret * a % m;
14         a = a * a % m, p >>= 1;
15     } return ret;
16 }

```

### 4.2 Fib

```

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

```

### 4.3 JosephusProblem

```

1 // asking last one O(klog(n))
2 // 編號從 1 開始, 結果要加 1
3 int josephus(int n, int k) {

```

```

4  if (k == 1) return n - 1;
5  int ans = 0;
6  for (int i = 2; i <= n; ) {
7      if (ans + k >= i) {
8          ans = (ans + k) % i;
9          i++;
10         continue;
11     }
12     int step = (i - 1 - ans - 1) / (k - 1); // 向下取
        整
13     if (i + step > n) {
14         ans += (n - (i - 1)) * k;
15         break;
16     }
17     i += step;
18     ans += step * k;
19 }
20 return ans;
21 }
22
23 // Josephus Problem
24 LL J(LL n, const vector<LL>& cs) {
25     LL ans = 0, m = cs.size(), sum = 0;
26     for (LL v : cs) sum += v;
27     if (n == 1 || sum == m) return n - 1;
28     for (LL i = 2; i <= n; ) {
29         LL d = min((n - i + 1) / m, (i - 2 - ans) / (sum - m)
30             );
31         if (d <= 0) {
32             ans = (ans + cs[(n - i) % m]) % i;
33             i++;
34             continue;
35         }
36         i += d * m, ans += d * sum;
37     }
38     return ans;
39 }
40 LL slow(LL n, const vector<LL>& cs, LL sum = 0) {
41     LL ans = 0;
42     for (LL i = 1; i <= n; i++) ans = (ans + cs[(n - i) % cs.
43         size()]) % i;
44     return ans;
45 }
46 int main() {
47     LL n, m;
48     while (cin >> n >> m) {
49         LL sum = 0;
50         vector<LL> cs(m);
51         for (auto& v : cs) {
52             cin >> v;
53             sum += v;
54         }
55         cout << J(n, cs) + 1 << endl;
56         //cout << slow(n, cs, sum) + 1 << endl;
57     }
58 }

```

## 4.4 SG

1 Anti Nim (取走最後一個石子者敗) :  
2 先手必勝 if and only if  
3 1. 「所有」堆的石子數都為 1 且遊戲的 SG 值為 0

4 2. 「有些」堆的石子數大於 1 且遊戲的 SG 值不為  
 0。  
5 Anti-SG (決策集合為空的遊戲者贏) :  
6 定義 SG 值為 0 時，遊戲結束，  
7 則先手必勝 if and only if  
8 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。  
9 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。  
10 -----  
11 Sprague -Grundy :  
12 1. 雙人、回合制  
13 2. 資訊完全公開  
14 3. 無隨機因素  
15 4. 可在有限步內結束  
16 5. 沒有和局  
17 6. 雙方可採取的行動相同  
18  
19 SG(S) 的值為 0 : 後手 (P) 必勝  
20 不為 0 : 先手 (N) 必勝

```

21
22 int mex(set S) {
23     // find the min number >= 0 that not in the S
24     // e.g. S = {0, 1, 3, 4} mex(S) = 2
25 }
26 state = []
27 int SG(A) {
28     if (A not in state) {
29         S = sub_states(A)
30         if (len(S) > 1) state[A] = reduce(operator.xor, [SG(
31             B) for B in S])
32         else state[A] = mex(set(SG(B) for B in next_states(A)
33             ))
34     }
35     return state[A]
36 }

```

## 4.5 線篩

```

1 void sieve(int n) {
2     vector<bool> p(n, true);
3     vector<int> prime;
4     for (int i = 2; i < n; i++) {
5         if (p[i]) prime.push_back(i);
6         for (auto& k : prime) {
7             if (i * k >= n) break;
8             p[i * k] = false;
9             if (i % k == 0) break;
10        }
11    }
12 }

```

## 4.6 鞋帶

```

1 vector<pair<int, int>> v(n);
2 for(auto&n:v) cin >> n.first >> n.second;
3 int area = 0;
4 for(int i = 0; i < v.size(); i++){
5     area += v[i].first * v[(i+1)%v.size()].second;
6     area -= v[i].second * v[(i+1)%v.size()].first;
7 }
8 cout << abs(area)/2 << endl;

```

## 4.7 質因數分解

```

1 LL func(const LL n, const LL mod, const int c) {
2     return (n * n % mod + c + mod) % mod;
3 }
4 LL pollorrho(const LL n, const int c) {
5     LL a = 1, b = 1;
6     a = func(a, n, c) % n;
7     b = func(b, n, c) % n;
8     b = func(b, n, c) % n;
9     while (__gcd(abs(a - b), n) == 1) {
10        a = func(a, n, c) % n;
11        b = func(b, n, c) % n;
12        b = func(b, n, c) % n;
13    }
14    return __gcd(abs(a - b), n);
15 }
16 void prefactor(LL &n, vector<LL> &v, const vector<int> &prime
17 ) {
18     for (int i = 0; i < prime.size(); ++i) {
19         while (n % prime[i] == 0) {
20             v.push_back(prime[i]);
21             n /= prime[i];
22         }
23     }
24 }
25 void smallfactor(LL n, vector<LL> &v, const vector<int> &isp,
26     const int MAXPRIME, const vector<int> &prime) {
27     if (n < MAXPRIME) {
28         while (isp[(int)n]) {
29             v.push_back(isp[(int)n]);
30             n /= isp[(int)n];
31         }
32     }
33     else {
34         for (int i = 0; i < prime.size() && prime[i] * prime[
35             i] <= n; ++i) {
36             while (n % prime[i] == 0) {
37                 v.push_back(prime[i]);
38                 n /= prime[i];
39             }
40         }
41         if (n != 1) v.push_back(n);
42     }
43 }
44 void comfactor(const LL &n, vector<LL> &v, const vector<int>
45     &prime, const vector<int> &isp, const int MAXPRIME) {
46     if (n < 1e9) {
47         smallfactor(n, v, isp, MAXPRIME, prime);
48         return;
49     }
50     if (Isprime(n)) {
51         v.push_back(n);
52         return;
53     }
54     LL d;
55     for (int c = 3; ++c) {
56         d = pollorrho(n, c);
57         if (d != n) break;
58     }
59     comfactor(d, v, prime, isp, MAXPRIME);
60     comfactor(n / d, v, prime, isp, MAXPRIME);
61 }
62 void factor(const LL &x, vector<LL> &v, const vector<int> &
63     prime, const vector<int> &isp, const int MAXPRIME) {
64     LL n = x;

```



```

60 if (n == 1) {
61     puts("Factor 1");
62     return;
63 }
64 prefactor(n, v, prime);
65 if (n == 1) return;
66 comfactor(n, v, prime, isp, MAXPRIME);
67 sort(v.begin(), v.end());
68 }
69
70 void AllFactor(const LL &n, vector<LL> &v, const vector<int>
71 &prime, const vector<int> &isp, const int MAXPRIME) {
72     vector<LL> tmp;
73     Factor(n, tmp, prime, isp, MAXPRIME);
74     v.clear();
75     v.push_back(1);
76     int len;
77     LL now = 1;
78     for (int i = 0; i < tmp.size(); ++i) {
79         if (i == 0 || tmp[i] != tmp[i - 1]) {
80             len = v.size();
81             now = 1;
82             *now = tmp[i];
83             for (int j = 0; j < len; ++j)
84                 v.push_back(v[j] * now);
85         }
86     }

```

## 4.8 擴展歐幾里德

```

1 // 給 a,b , 解 ax+by=gcd(a,b)
2 typedef pair<ll, ll> pii;
3 pii extgcd(ll a, ll b) {
4     if (b == 0) return {1, 0};
5     ll k = a / b;
6     pii p = extgcd(b, a - k * b);
7     return {p.second, p.first - k * p.second};
8 }

```

## 5 Other

### 5.1 精度

```

1 from decimal import*
2 getcontext().prec = 1000000
3 n = Decimal(input())

```

### 5.2 莫隊

```

1 // 區間眾數
2 #define ql first.first
3 #define qr first.second
4 #define id second
5 int N, Q, K;
6 vector<int> A;
7 vector<pair<pii, int>> qrys;
8 ll ans[200007];
9 void init() {
10     cin >> N >> Q;
11     K = max(1, (int)sqrt(Q)); // Size of the block
12     A.clear();
13     A.resize(N);
14     qrys.clear();

```

```

15     qrys.resize(Q);
16
17     for (auto &i : A) cin >> i;
18     int cnt = 0;
19     for (auto &i : qrys) {
20         cin >> i.ql >> i.qr;
21         --i.ql;
22         --i.qr; // Convert to 0-base
23         i.id = cnt++;
24     }
25 }
26 void solve() {
27     sort(qrys.begin(), qrys.end(),
28         [&](const pair<pii, int> &a, const pair<pii, int> &b) {
29             {
30                 if (a.ql / K == b.ql / K) // Same left block,
31                     sort by right block
32                     return a.qr < b.qr;
33                 return a.ql / K < b.ql / K; // Otherwise, sort by
34                     left block
35             });
36     int l = 0, r = -1; // Initial answer window
37     ll sum = 0;
38     for (auto &i : qrys) {
39         while (l > i.ql) { l--; sum += A[l]; } // Extend left
40         boundary
41         while (r < i.qr) { r++; sum += A[r]; } // Extend
42         right boundary
43         while (l < i.ql) { sum -= A[l]; l++; } // Shrink left
44         boundary
45         while (r > i.qr) { sum -= A[r]; r--; } // Shrink
46         right boundary
47         ans[i.id] = sum;
48     }
49     for (int i = 0; i < Q; i++) cout << ans[i] << endl;
50 }
51 int main() {
52     init();
53     solve();
54     return 0;
55 }

```

### 5.3 離散化

```

1 vector<int> v(1000), b(1000);
2 for(auto&n:v)cin>>n;
3 sort(v.begin(),v.end());
4 auto len = unique(v.begin(),v.end())-v.begin();
5 v.resize(len);
6 for(int i = 0; i < v.size(); i++){
7     b[i] = lower_bound(v.begin(),v.end(),v[i])-v.begin();
8 }

```

## 6 String

### 6.1 ACAutomaton

```

1 // Definition of the Aho-Corasick Trie structure
2 struct _AC {
3     _AC* child[26];
4     _AC* Fail;
5     vector<pair<int, int>> out;
6     _AC() {
7         Fail = NULL;

```

```

8         memset(child, 0, sizeof(child));
9     }
10 } *root;
11 // Function to insert a pattern into the Aho-Corasick Trie
12 void Insert_AC(string s) {
13     int n;
14     _AC* p = root;
15     for (auto& k : s) {
16         n = k - 'a';
17         if (!p->child[n]) p->child[n] = new _AC();
18         p = p->child[n];
19     }
20     p->out.push_back({ s.size(), 0 });
21 }
22 // Function to construct the Aho-Corasick Trie with failure
23 // links
24 void Construct_AC() {
25     queue<_AC*> Q;
26     for (int k = 0; k < 26; k++) {
27         if (root->child[k]) {
28             root->child[k]->Fail = root;
29             Q.push(root->child[k]);
30         } else {
31             root->child[k] = root;
32         }
33     }
34     _AC* p;
35     while (!Q.empty()) {
36         p = Q.front();
37         Q.pop();
38         for (int k = 0; k < 26; k++) {
39             if (!p->child[k]) {
40                 p->child[k] = p->Fail->child[k];
41             } else {
42                 p->child[k]->Fail = p->Fail->child[k];
43                 for (auto& i : p->Fail->child[k]->out)
44                     p->child[k]->out.push_back({ i.first, 1
45 });
46                 Q.push(p->child[k]);
47             }
48         }
49     }
50 // Function to match patterns in the given text using the Aho
51 // -Corasick Trie
52 void Match_AC(string t) {
53     int n;
54     _AC* p = root;
55     for (int k = 0; k < t.size(); k++) {
56         n = t[k] - 'a';
57         p = p->child[n];
58         _AC* fail = p;
59         while (fail != root && fail->out.size()) {
60             for (auto& i : fail->out)
61                 if (!i.second || fail->Fail->out.size())
62                     cout << t.substr(k - i.first + 1, i.first
63 ) << '\n';
64             fail->out.clear();
65             fail->Fail->out.clear();
66             fail = fail->Fail;
67         }
68     }
69 }
70 // Main function to test the Aho-Corasick Trie
71 int main() {
72     int n, m;

```

```

70 string p, t;
71 while (cin >> n >> m) {
72     root = new AC();
73     while (n--){
74         cin >> p;
75         Insert_AC(p);
76     }
77     Construct_AC();
78     cin >> t;
79     Match_AC(t);
80 }
81 }

```

## 6.2 EditDistance

```

1  ll edd(string& src, string& dst, ll ins, ll del, ll sst) {
2      vector<vector<long long>> dp(src.size() + 1, vector<long
3          long>(dst.size() + 1));
4      ll dp[src.size() + 1][dst.size() + 1];
5      for (int i = 0; i <= src.size(); i++) {
6          for (int j = 0; j <= dst.size(); j++) {
7              if (i == 0) dp[i][j] = ins * j;
8              else if (j == 0) dp[i][j] = del * i;
9              else if (src[i - 1] == dst[j - 1]) dp[i][j] = dp[
10                 i - 1][j - 1];
11              else dp[i][j] = min(dp[i][j - 1] + ins, min(dp[i -
12                 1][j] + del, dp[i - 1][j - 1] + sst));
13          }
14      }
15      return dp[src.size()][dst.size()];
16  }

```

## 6.3 KMP

```

1  // KMP fail function.
2  int* kmp_fail(string& s) {
3      int* f = new int[s.size()];
4      int p = f[0] = -1;
5      for (int i = 1; i < s.size(); i++) {
6          while (p != -1 && s[p + 1] != s[i]) p = f[p];
7          if (s[p + 1] == s[i]) p++;
8          f[i] = p;
9      }
10     return f;
11 }
12 // Function to count how many times sub appears in str.
13 int kmp_count(string& str, string& sub) {
14     int* fail = kmp_fail(sub);
15     int p = -1, ret = 0;
16     for (int i = 0; i < str.size(); i++) {
17         while (p != -1 && sub[p + 1] != str[i]) p = fail[p];
18         if (sub[p + 1] == str[i]) p++;
19         if (p == sub.size() - 1) p = fail[p], ret++;
20     }
21     delete[] fail;
22     return ret;
23 }
24 }
25 //問sub在str第一次出現的開頭index。-1表示找不到。
26 int kmp(string& str, string& sub) {
27     int* fail = kmp_fail(sub);
28     int i = 0, j = 0;
29     while (i < str.size() && j < sub.size()) {
30         if (sub[j] == str[i]) i++, j++;

```

```

32         else if (j == 0) i++;
33         else j = fail[j - 1] + 1;
34     }
35     delete[] fail;
36     return j == sub.size() ? (i - j) : -1;
37 }

```

## 6.4 LPS

```

1  int lps_length(string s) {
2      int N = 2 * s.size() + 1;
3      vector<int> dp(N);
4      string s2 = "*";
5      for (auto& c : s) s2.push_back(c), s2.push_back('*');
6      int C = 0, R = 0;
7      for (int i = 1; i < N; i++) {
8          if (i > R) C = R = i;
9          else {
10             int mirrorI = 2 * C - i;
11             dp[i] = min(dp[mirrorI], R - i);
12         }
13         int j = dp[i] + 1;
14         while ((i - j) >= 0 && (i + j < N) && (s2[i - j] ==
15             s2[i + j])) j++;
16         dp[i] = j - 1;
17         if (i + dp[i] > R) C = i, R = i + dp[i];
18     }
19     return *max_element(dp.begin(), dp.end());
20 }
21 string lps(string s) {
22     int N = 2 * s.size() + 1;
23     vector<int> dp(N);
24     string s2 = "*";
25     for (auto& c : s) s2.push_back(c), s2.push_back('*');
26     int C = 0, R = 0;
27     for (int i = 1; i < N; i++) {
28         if (i > R) C = R = i;
29         else {
30             int mirrorI = 2 * C - i;
31             dp[i] = min(dp[mirrorI], R - i);
32         }
33         int j = dp[i] + 1;
34         while ((i - j) >= 0 && (i + j < N) && (s2[i - j] ==
35             s2[i + j])) j++;
36         dp[i] = j - 1;
37         if (i + dp[i] > R) C = i, R = i + dp[i];
38     }
39     auto it = max_element(dp.begin(), dp.end());
40     int maxLen = *it;
41     int index = it - dp.begin();
42     return s.substr((index - maxLen) / 2, maxLen);
43 }

```

## 6.5 Trie

```

1  class Trie {
2  private:
3      struct Node {
4          int cnt = 0, sum = 0;
5          Node* tr[128] = {}; // Array of pointers to child
6              nodes
7      };
8      ~Node() {
9          for (int i = 0; i < 128; i++) {

```

```

9             if (tr[i]) delete tr[i];
10         }
11     };
12 };
13
14 Node* root;
15
16 public:
17     // Constructor
18     Trie() {
19         root = new Node();
20     }
21     // Destructor
22     ~Trie() {
23         delete root;
24     }
25     // Function to insert a string into the Trie
26     void insert(char* s) {
27         Node* ptr = root;
28         for (; *s; s++) {
29             if (!ptr->tr[*s]) {
30                 ptr->tr[*s] = new Node();
31             }
32             ptr = ptr->tr[*s];
33             ptr->sum++;
34         }
35         ptr->cnt++;
36     }
37     // Function to count the occurrences of a string in the
38     // Trie
39     inline int count(char* s) {
40         Node* ptr = find(s);
41         return ptr ? ptr->cnt : 0;
42     }
43     // Function to find a node corresponding to a string
44     Node* find(char* s) {
45         Node* ptr = root;
46         for (; *s; s++) {
47             if (!ptr->tr[*s]) return nullptr;
48             ptr = ptr->tr[*s];
49         }
50         return ptr;
51     }
52     // Function to erase a string from the Trie
53     bool erase(char* s) {
54         if (!count(s)) return false; // If the word doesn't
55         // exist, return false
56         Node* ptr = root;
57         for (char* t = s; *t; t++) {
58             Node* tmp = ptr->tr[*t];
59             tmp->sum--;
60             if (tmp->sum == 0) {
61                 delete tmp;
62                 ptr->tr[*t] = nullptr;
63                 return true;
64             }
65             ptr = tmp;
66         }
67         ptr->cnt--; // Decrease the count of the word itself
68         return true;
69     };

```

# MCU-NL CODEBOOK

## Contents

<b>1 Data Structure</b>	<b>1</b>	<b>3 Geometry</b>	<b>5</b>	<b>5 Other</b>	<b>9</b>
1.1 CDQ	1	3.1 ClosestPair	5	5.1 精度	9
1.2 DSU	1	3.2 ConvexHull	5	5.2 莫隊	9
1.3 Dag	2	3.3 EulerCircuit	6	5.3 離散化	9
1.4 Kruskal	2	3.4 Hopcroft-Karp	6	<b>6 String</b>	<b>9</b>
1.5 Prim	2	3.5 Hungarian	6	6.1 ACAutomaton	9
1.6 SegmentTreeLazy	2	3.6 MinCircle	6	6.2 EditDistance	10
1.7 SegmentTree	3	3.7 Angle	6	6.3 KMP	10
<b>2 DP</b>	<b>3</b>	3.8 旋轉卡尺	7	6.4 LPS	10
2.1 BellmanFord	3	3.9 LCA	7	6.5 Trie	10
2.2 EdmondsKarp	3	3.10 SPFA	7		
2.3 LCS	4	<b>4 Math</b>	<b>7</b>		
2.4 LCS2LIS	4	4.1 FPow	7		
		2.5 LCS3D	4	4.2 Fib	7
		2.6 LIS	4	4.3 JosephusProblem	7
		2.7 MaxFlow	4	4.4 SG	8
		2.8 SPFA	5	4.5 線節	8
		2.9 DP	5	4.6 鞋帶	8
				4.7 質因數分解	8
				4.8 擴展歐幾里德	9