

Contents

1 DataStructure

- 1.1 Treap
- 1.2 Dynamic Segment Tree

2 Math

- 2.1 Mu
- 2.2 Lucas
- 2.3 Inv
- 2.4 Formula
 - 2.4.1 Dirichlet Convolution
 - 2.4.2 Burnside's Lemma
 - 2.4.3 Pick Theorem

3 String

- 3.1 KMP
- 3.2 Longest Palindrome

4 Graph

- 4.1 one-out-degree (CSES Planets Cycles)
- 4.2 Dijkstra
- 4.3 MaximumFlow
- 4.4 SCC
- 4.5 2-SAT(CSES Giant Pizza)

5 DP

- 5.1 Li-Chao Segment Tree
- 5.2 CHO

6 Geometry

- 6.1 Intersect
- 6.2 Inside
- 6.3 Minimum Euclidean Distance

1. DataStructure

1.1. Treap

```

1 #define pii pair<int, int>
2 struct node {
3     int tag = 0;
4     int sum = 0;
5     int prio = rand();
6     int lson = 0;
7     int rson = 0;
8     int si = 0;
9     int val = 0;
10 };
11 node treap[400005];
12 int cnt = 0;
13 int root = 0;
14
15 void update(int index) {
16     int lson = treap[index].lson;
17     int rson = treap[index].rson;
18     treap[index].si = treap[lson].si + treap[rson].si + 1;
19     treap[index].sum = treap[lson].sum +
20     treap[rson].sum + treap[index].val;
21 }
22
23 void push(int index) {
24     if (!treap[index].tag)
25         return;
26     swap(treap[index].lson, treap[index].rson);
27     int lson = treap[index].lson;
28     int rson = treap[index].rson;
29     treap[lson].tag ^= 1;
30     treap[rson].tag ^= 1;
31     treap[index].tag = 0;
32 }
33
34 pii split(int rk, int index) {
35     if (!index)
36         return {0, 0};
37     push(index);
38     int lson = treap[index].lson;
39     int rson = treap[index].rson;
40     if (rk <= treap[lson].si) {
41         pii temp = split(rk, lson);
42         treap[index].lson = temp.second;
43         update(index);
44         return {temp.first, index};
45     } else {
46         pii temp = split(rk - treap[lson].si - 1, rson);

```

```

47     treap[index].rson = temp.first;
48     update(index);
49     return {index, temp.second};
50 }
51
52
53 int merge(int x, int y) {
54     if (!x && !y)
55         return 0;
56     if (!x && y)
57         return y;
58     if (x && !y)
59         return x;
60     push(x);
61     push(y);
62     if (treap[x].prio < treap[y].prio) {
63         treap[x].rson = merge(treap[x].rson, y);
64         update(x);
65         return x;
66     } else {
67         treap[y].lson = merge(x, treap[y].lson);
68         update(y);
69         return y;
70     }
71 }
72
73 void insert(int x, int v) {
74     pii temp = split(x - 1, root);
75     cnt++;
76     treap[cnt].val = v;
77     update(cnt);
78     temp.first = merge(temp.first, cnt);
79     root = merge(temp.first, temp.second);
80 }
81
82 int query(int l, int r) {
83     pii R = split(r, root);
84     pii L = split(l - 1, R.first);
85     int ret = treap[L.second].sum;
86     R.first = merge(L.first, L.second);
87     root = merge(R.first, R.second);
88     return ret;
89 }
90
91 void modify(int l, int r) {
92     pii R = split(r, root);
93     pii L = split(l - 1, R.first);
94     treap[L.second].tag ^= 1;
95     R.first = merge(L.first, L.second);
96     root = merge(R.first, R.second);
97 }

```

1.2. Dynamic Segment Tree

```

1 #define int long long
2 using namespace std;
3
4 int n, q;
5 struct node {
6     int data, lson, rson, tag;
7     int rv() { return data + tag; }
8 };
9
10 node tree[20000005];
11 int a[200005];
12 int now = 1;
13 int mx = 1000000005;
14
15 void push(int index) {
16     if (!tree[index].lson)
17         tree[index].lson = ++now;
18     if (!tree[index].rson)
19         tree[index].rson = ++now;
20 }
21
22 int lson = tree[index].lson;
23 int rson = tree[index].rson;
24 tree[lson].tag += tree[index].tag;
25 tree[rson].tag += tree[index].tag;
26 tree[index].data = tree[index].rv();
27 tree[index].tag = 0;
28 }
29
30 void modify(int l, int r, int L, int R, int val, int index) {
31     if (l == L && r == R) {
32         tree[index].tag += val;
33         return;
34     }
35     int mid = (l + r) >> 1;

```

```

37 push(index);
38 int lson = tree[index].lson;
39 int rson = tree[index].rson;
40 if (R <= mid) {
41     modify(l, mid, L, R, val, lson);
42 } else if (L > mid) {
43     modify(mid + 1, r, L, R, val, rson);
44 } else {
45     modify(l, mid, L, mid, val, lson);
46     modify(mid + 1, r, mid + 1, R, val, rson);
47 }
48 tree[index].data = tree[lson].rv() + tree[rson].rv();
49 }

51 int query(int l, int r, int L, int R, int index) {
52     // cout << L << " " << R << "\n";
53     if (l == L && r == R) {
54         return tree[index].rv();
55     }
56     int mid = (l + r) >> 1;
57     push(index);
58     int lson = tree[index].lson;
59     int rson = tree[index].rson;
60     if (R <= mid) {
61         return query(l, mid, L, R, lson);
62     }
63     if (L > mid) {
64         return query(mid + 1, r, L, R, rson);
65     }
66     return query(l, mid, L, mid, lson) + query(mid + 1, r, mid + 1, R, rson);
67 }

69 signed main() {
70     ios::sync_with_stdio(0);
71     cin.tie(0);
72     cout.tie(0);
73     cin >> n >> q;
74     for (int i = 1; i <= n; i++) {
75         cin >> a[i];
76         modify(1, mx, a[i], a[i], 1, 1);
77     }
78     while (q--) {
79         char mode;
80         int x, y;
81         cin >> mode;
82         if (mode == '?') {
83             cin >> x >> y;
84             cout << query(1, mx, x, y, 1) << "\n";
85         } else {
86             cin >> x >> y;
87             modify(1, mx, a[x], a[x], -1, 1);
88             a[x] = y;
89             modify(1, mx, a[x], a[x], 1, 1);
90         }
91     }
92 }

```

2. Math

2.1. Mu

```

1 vector<int> prime;
2 bitset<1000005> vis;
3 int n;
4 int mu[1000005];
5
6 void init() {
7     for (int i = 2; i <= n; i++) {
8         if (!vis[i]) {
9             prime.push_back(i);
10            mu[i] = -1;
11        }
12        for (int p : prime) {
13            if (i * p > n) break;
14            vis[i * p] = 1;
15            if (i % p == 0) {
16                mu[i * p] = 0;
17                break;
18            } else {
19                mu[i * p] = mu[i] * mu[p];
20            }
21        }
22    }
23 }

```

2.2. Lucas

```

1 int fact[100005];
2 int p;

```

```

3 void init() {
4     fact[0] = 1;
5     for (int i = 1; i <= p; i++) {
6         fact[i] = fact[i - 1] * i % p;
7     }
8 }
9
10 int inv(int x, int p) {
11     if (x == 1) return 1;
12     return (p - p / x) * inv(p % x, p) % p;
13 }
14
15 int c(int x, int y, int p) {
16     if (x < y) return 0;
17     int k = fact[x] * inv(fact[y], p) % p;
18     return k * inv(fact[x - y], p) % p;
19 }
20
21 int lucas(int x, int y, int p) {
22     if (x == 0) return 1;
23     return lucas(x / p, y / p, p) % p * c(x % p, y % p, p) % p;
24 }

```

2.3. Inv

```

1 int exgcd(int a, int b, int &x, int &y) {
2     if (b == 0) {
3         x = 1;
4         y = 0;
5         return a;
6     }
7     int d = exgcd(b, a % b, y, x);
8     y -= x * (a / b);
9     return d;
10 }
11
12 int inv(int a, int p) {
13     int x, y;
14     exgcd(a, p, x, y);
15     return (x % p + p) % p;
16 }

```

2.4. Formula

2.4.1. Dirichlet Convolution

$$\varepsilon = \mu * 1$$

$$\varphi = \mu * \text{Id}$$

2.4.2. Burnside's Lemma

Let X be a set and G be a group that acts on X . For $g \in G$, denote by X^g the elements fixed by g :

$$X^g = \{x \in X \mid gx \in X\}$$

Then

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|.$$

2.4.3. Pick Theorem

$$A = i + \frac{b}{2} - 1$$

3. String

3.1. KMP

```

1 string s, t;
2 int pmt[1000005];
3
4 void init() {
5     for (int i = 1, j = 0; i < t.size(); i++) {
6         while (j && t[j] != t[i]) {
7             j = pmt[j - 1];
8         }
9         if (t[j] == t[i]) j++;
10        pmt[i] = j;
11    }
12 }
13
14 int kmp(string s) {
15     int ret = 0;
16     for (int i = 0, j = 0; i < s.size(); i++) {

```

```

19     while (j && s[i] ^ t[j]) {
20         j = pmt[j - 1];
21     }
22     if (s[i] == t[j]) {
23         j++;
24     }
25     if (j == t.size()) {
26         ret++;
27         j = pmt[j - 1];
28     }
29     return ret;
30 }

```

3.2. Longest Palindrome

```

1  #define int long long
2  using namespace std;
3
4  string s;
5  string t;
6  int n;
7  int d[2000005];
8  int ans = 0;
9
10 signed main() {
11     cin >> t;
12     n = t.size();
13     for (int i = 0; i < 2 * n + 1; i++) {
14         if (i & 1 ^ 1) {
15             s += '0';
16         } else {
17             s += t[i / 2];
18         }
19     }
20     n = s.size();
21     d[0] = 1;
22     for (int i = 0, l = 0, r = 0; i < n; i++) {
23         if (i > r) {
24             d[i] = 1;
25             bool a = i + d[i] < n;
26             bool b = i - d[i] >= 0;
27             bool c = (s[i + d[i]] == s[i - d[i]]);
28             while (a && b && c) {
29                 d[i]++;
30                 a = i + d[i] < n;
31                 b = i - d[i] >= 0;
32                 c = (s[i + d[i]] == s[i - d[i]]);
33             }
34             l = i - d[i] + 1;
35             r = i + d[i] - 1;
36         } else {
37             int j = l + r - i;
38             if (j - d[j] + 1 > l) {
39                 d[i] = d[j];
40             } else {
41                 d[i] = r - i + 1;
42                 a = i + d[i] < n;
43                 b = i - d[i] >= 0;
44                 c = (s[i + d[i]] == s[i - d[i]]);
45                 while (a && b && c) {
46                     d[i]++;
47                     a = i + d[i] < n;
48                     b = i - d[i] >= 0;
49                     c = (s[i + d[i]] == s[i - d[i]]);
50                 }
51                 l = i - d[i] + 1;
52                 r = i + d[i] - 1;
53             }
54         }
55         // cout << d[i] << " ";
56         if (d[i] > d[ans]) {
57             ans = i;
58         }
59     }
60     for (int i = ans - d[ans] + 1; i < ans + d[ans]; i++) {
61         if (s[i] ^ '0') {
62             cout << s[i];
63         }
64     }
65 }

```

4. Graph

4.1. one-out-degree (CSES Planets Cycles)

```

1  #define int long long
2  using namespace std;

```

```

5  int n, q;
6  int a[200005];
7  int r[200005];
8  int d[200005];
9  int cycle[200005];
10 int len[200005];
11 int cnt = 0;
12 vector<int> v[200005];
13 bitset<200005> vis1;
14 bitset<200005> vis2;
15
16 void findcycle(int x) {
17     while (!vis1[x]) {
18         vis1[x] = 1;
19         x = a[x];
20     }
21     cnt++;
22     cycle[x] = cnt;
23     r[x] = 0;
24     len[cnt] = 1;
25     int temp = a[x];
26     while (temp ^ x) {
27         r[temp] = len[cnt];
28         len[cnt]++;
29         cycle[temp] = cnt;
30         temp = a[temp];
31     }
32 }
33
34 void dfs(int x) {
35     if (vis2[x])
36         return;
37     vis2[x] = 1;
38     for (int i : v[x]) {
39         dfs(i);
40     }
41 }
42
43 void dfs2(int x) {
44     if (cycle[x] || d[x])
45         return;
46     dfs2(a[x]);
47     d[x] = d[a[x]] + 1;
48     r[x] = r[a[x]];
49     cycle[x] = cycle[a[x]];
50 }
51
52 signed main() {
53     ios::sync_with_stdio(0);
54     cin.tie(0);
55     cout.tie(0);
56     cin >> n;
57     for (int i = 1; i <= n; i++) {
58         cin >> a[i];
59         v[i].push_back(a[i]);
60         v[a[i]].push_back(i);
61     }
62     for (int i = 1; i <= n; i++) {
63         if (!vis2[i]) {
64             findcycle(i);
65             dfs(i);
66         }
67     }
68     for (int i = 1; i <= n; i++) {
69         if (!cycle[i] && !r[i]) {
70             dfs2(i);
71         }
72     }
73     for (int i = 1; i <= n; i++) {
74         cout << d[i] + len[cycle[i]] << " ";
75     }
76 }

```

4.2. Dijkstra

```

1  int n, m;
2  vector<pair<int, int>> v[100005];
3  bitset<100005> vis;
4  int dis[100005];
5
6  void dijkstra(int x) {
7      priority_queue<pair<int, int>, vector<pair<int, int>>,
8          greater<pair<int, int>>>
9          pq;
10     memset(dis, 0x3f, sizeof(dis));
11     dis[x] = 0;
12     pq.push({0, x});
13     while (!pq.empty()) {
14         pair<int, int> now = pq.top();
15         pq.pop();

```

```

17     if (vis[now.second])
18         continue;
19     vis[now.second] = 1;
20     for (auto [i, w] : v[now.second]) {
21         if (vis[i])
22             continue;
23         if (dis[now.second] + w < dis[i]) {
24             dis[i] = dis[now.second] + w;
25             pq.push({dis[i], i});
26         }
27     }
28 }

```

4.3. MaximumFlow

```

1  #define int long long
2  using namespace std;
3
4  int n, m;
5  vector<int> v[1005];
6  int head[1005];
7  int c[1005][1005];
8  int lv[1005];
9  int ans = 0;
10
11 bool bfs() {
12     memset(head, 0, sizeof(head));
13     memset(lv, 0, sizeof(lv));
14     queue<int> q;
15     q.push(1);
16     while (!q.empty()) {
17         int now = q.front();
18         q.pop();
19         if (now == n)
20             continue;
21         for (int i : v[now]) {
22             if (i != 1 && c[now][i] && !lv[i]) {
23                 lv[i] = lv[now] + 1;
24                 q.push(i);
25             }
26         }
27     }
28     return lv[n];
29 }
30
31 int dfs(int x, int flow) {
32     int ret = 0;
33     if (x == n)
34         return flow;
35     for (int i = head[x]; i < v[x].size(); i++) {
36         int y = v[x][i];
37         head[x] = y;
38         if (c[x][y] && lv[y] == lv[x] + 1) {
39             int d = dfs(y, min(flow, c[x][y]));
40             flow -= d;
41             c[x][y] -= d;
42             c[y][x] += d;
43             ret += d;
44         }
45     }
46     return ret;
47 }
48
49 signed main() {
50     cin >> n >> m;
51     while (m--) {
52         int x, y, z;
53         cin >> x >> y >> z;
54         if (c[x][y] || c[y][x]) {
55             c[x][y] += z;
56             continue;
57         }
58         v[x].push_back(y);
59         v[y].push_back(x);
60         c[x][y] = z;
61     }
62     while (bfs()) {
63         ans += dfs(1, INT_MAX);
64     }
65     cout << ans;
66 }

```

4.4. SCC

```

1  int n, m;
2  vector<int> v[100005];
3  int d[100005];
4  int low[100005];

```

```

5  int cnt = 0;
6  stack<int> s;
7  int scc[100005];
8  int now = 0;
9
10 void dfs(int x) {
11     d[x] = low[x] = ++cnt;
12     s.push(x);
13     for (int i : v[x]) {
14         if (scc[i])
15             continue;
16         if (d[i]) {
17             low[x] = min(low[x], d[i]);
18         } else {
19             dfs(i);
20             low[x] = min(low[x], low[i]);
21         }
22     }
23     if (d[x] == low[x]) {
24         now++;
25         while (!s.empty()) {
26             int k = s.top();
27             s.pop();
28             scc[k] = now;
29             if (k == x)
30                 break;
31         }
32     }
33 }

```

4.5. 2-SAT(CSES Giant Pizza)

```

1  #define int long long
2  using namespace std;
3
4  int n, m;
5  vector<int> v[200005];
6  int d[200005];
7  int low[200005];
8  int cnt = 0;
9  int now = 0;
10 int scc[200005];
11 stack<int> s;
12 int op[200005];
13 vector<int> v2[200005];
14 int ind[200005];
15 queue<int> q;
16 int ans[200005];
17
18 int no(int x) {
19     if (x > m)
20         return x - m;
21     return x + m;
22 }
23
24 void dfs(int x) {
25     d[x] = low[x] = ++cnt;
26     s.push(x);
27     for (int i : v[x]) {
28         if (scc[i])
29             continue;
30         if (d[i]) {
31             low[x] = min(low[x], d[i]);
32         } else {
33             dfs(i);
34             low[x] = min(low[x], low[i]);
35         }
36     }
37     if (d[x] == low[x]) {
38         now++;
39         while (!s.empty()) {
40             int k = s.top();
41             s.pop();
42             scc[k] = now;
43             if (k == x)
44                 break;
45         }
46     }
47 }
48
49 signed main() {
50     ios::sync_with_stdio(0);
51     cin.tie(0);
52     cout.tie(0);
53     cin >> n >> m;
54     while (n--) {
55         char a, b;
56         int x, y;
57         cin >> a >> x >> b >> y;
58         if (a == '-')

```

```

    x = no(x);
    if (b == '-')
        y = no(y);
    v[no(x)].push_back(y);
    v[no(y)].push_back(x);
}
for (int i = 1; i <= 2 * m; i++) {
    if (!d[i]) {
        dfs(i);
    }
}
for (int i = 1; i <= m; i++) {
    if (scc[i] ^ scc[i + m]) {
        op[scc[i]] = scc[i + m];
        op[scc[i + m]] = scc[i];
    } else {
        cout << "IMPOSSIBLE";
        exit(0);
    }
}
for (int i = 1; i <= 2 * m; i++) {
    for (int j : v[i]) {
        if (scc[i] ^ scc[j]) {
            v2[scc[j]].push_back(scc[i]);
            ind[scc[i]]++;
        }
    }
}
for (int i = 1; i <= now; i++) {
    if (!ind[i]) {
        q.push(i);
    }
}
while (!q.empty()) {
    int k = q.front();
    q.pop();
    if (!ans[k]) {
        ans[k] = 1;
        ans[op[k]] = 2;
    }
    for (int i : v2[k]) {
        ind[i]--;
        if (!ind[i]) {
            q.push(i);
        }
    }
}
for (int i = 1; i <= m; i++) {
    if (ans[scc[i]] == 1) {
        cout << "+ ";
    } else {
        cout << "- ";
    }
}
}

```

5. DP

5.1. Li-Chao Segment Tree

```

1 struct line {
2     int a, b = 1000000000000000;
3     int y(int x) { return a * x + b; }
4 };
5
6 line tree[4000005];
7 int n, x;
8 int s[200005];
9 int f[200005];
10 int dp[200005];
11
12 void update(line ins, int l = 1, int r = 1e6, int index = 1) {
13     if (l == r) {
14         if (ins.y(l) < tree[index].y(l)) {
15             tree[index] = ins;
16         }
17         return;
18     }
19     int mid = (l + r) >> 1;
20     if (tree[index].a < ins.a)
21         swap(tree[index], ins);
22     if (tree[index].y(mid) > ins.y(mid)) {
23         swap(tree[index], ins);
24         update(ins, l, mid, index << 1);
25     } else {
26         update(ins, mid + 1, r, index << 1 | 1);
27     }
28 }
29

```

```

int query(int x, int l = 1, int r = 1000000, int index = 1) {
    int cur = tree[index].y(x);
    if (l == r) {
        return cur;
    }
    int mid = (l + r) >> 1;
    if (x <= mid) {
        return min(cur, query(x, l, mid, index << 1));
    } else {
        return min(cur, query(x, mid + 1, r, index << 1 | 1));
    }
}

```

5.2. CHO

```

1 struct line {
2     int a, b;
3     int y(int x) { return a * x + b; }
4 };
5
6 struct CHO {
7     deque<line> dq;
8     int intersect(line x, line y) {
9         int d1 = x.b - y.b;
10        int d2 = y.a - x.a;
11        return d1 / d2;
12    }
13    bool check(line x, line y, line z) {
14        int I12 = intersect(x, y);
15        int I23 = intersect(y, z);
16        return I12 < I23;
17    }
18    void insert(int a, int b) {
19        if (!dq.empty() && a == dq.back().a)
20            return;
21        while (dq.size() >= 2 &&
22            !check(dq[dq.size() - 2], dq[dq.size() - 1], {a, b})) {
23            dq.pop_back();
24        }
25        dq.push_back({a, b});
26    }
27    void update(int x) {
28        while (dq.size() >= 2 && dq[0].y(x) >= dq[1].y(x)) {
29            dq.pop_front();
30        }
31    }
32    int query(int x) {
33        update(x);
34        return dq.front().y(x);
35    }
36 };

```

6. Geometry

6.1. Intersect

```

1 struct point {
2     int x, y;
3     point operator+(point b) { return {x + b.x, y + b.y}; }
4     point operator-(point b) { return {x - b.x, y - b.y}; }
5     int operator*(point b) { return x * b.x + y * b.y; }
6     int operator^(point b) { return x * b.y - y * b.x; }
7 };
8
9 bool onseg(point x, point y, point z) {
10     return ((x - z) ^ (y - z)) == 0 && (x - z) * (y - z) <= 0;
11 }
12
13 int dir(point x, point y) {
14     int k = x ^ y;
15     if (k == 0)
16         return 0;
17     if (k > 0)
18         return 1;
19     return -1;
20 }
21
22 bool intersect(point x, point y, point z, point w) {
23     if (onseg(x, y, z) || onseg(x, y, w))
24         return 1;
25     if (onseg(z, w, x) || onseg(z, w, y))
26         return 1;
27     if (dir(y - x, z - x) * dir(y - x, w - x) == -1 &&
28         dir(z - w, x - w) * dir(z - w, y - w) == -1) {
29         return 1;
30     }
31     return 0;
32 }

```

6.2. Inside

```

1 int inside(point p) {
2     int ans = 0;
3     for (int i = 1; i <= n; i++) {
4         if (onseg(a[i], a[i + 1], {p.x, p.y})) {
5             return -1;
6         }
7         if (intersect({p.x, p.y}, {INF, p.y}, a[i], a[i + 1])) {
8             ans ^= 1;
9         }
10        point temp = a[i].y > a[i + 1].y ? a[i] : a[i + 1];
11        if (temp.y == p.y && temp.x > p.x) {
12            ans ^= 1;
13        }
14    }
15    return ans;
16 }

```

6.3. Minimum Euclidean Distance

```

1 #define int long long
2 #define pii pair<int, int>
3 using namespace std;
4
5 int n;
6 vector<pair<int, int>> v;
7 set<pair<int, int>> s;
8 int dd = LONG_LONG_MAX;
9
10 int dis(pii x, pii y) {
11     return (x.first - y.first) * (x.first - y.first) +
12            (x.second - y.second) * (x.second - y.second);
13 }
14
15 signed main() {
16     ios::sync_with_stdio(0);
17     cin.tie(0);
18     cout.tie(0);
19     cin >> n;
20     for (int i = 0; i < n; i++) {
21         int x, y;
22         cin >> x >> y;
23         x += 1000000000;
24         v.push_back({x, y});
25     }
26     sort(v.begin(), v.end());
27     int l = 0;
28     for (int i = 0; i < n; i++) {
29         int d = ceil(sqrt(dd));
30         while (l < i && v[i].first - v[l].first > d) {
31             s.erase({v[l].second, v[l].first});
32             l++;
33         }
34         auto x = s.lower_bound({v[i].second - d, 0});
35         auto y = s.upper_bound({v[i].second + d, 0});
36         for (auto it = x; it != y; it++) {
37             dd = min(dd, dis({it->second, it->first}, v[i]));
38         }
39         s.insert({v[i].second, v[i].first});
40     }
41     cout << dd;
42 }

```