

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

7 Tree

- 7.1 Heavy Light Decomposition (modify and query on path)
- 7.2 LCA

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[index].sum += treap[rson].sum;
21     treap[index].sum += treap[index].val;
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 pii split(int rk, int index) {
34     if (!index)
35         return {0, 0};
36     push(index);
37     int lson = treap[index].lson;
38     int rson = treap[index].rson;
39     if (rk <= treap[lson].si) {
40         pii temp = split(rk, lson);
41         treap[index].lson = temp.second;

```

```

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

```

```

33     tree[index].tag += val;
34     return;
35 }
36 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 }

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

67 }

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

```

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
4 void init() {
5     fact[0] = 1;
6     for (int i = 1; i <= p; i++) {
7         fact[i] = fact[i - 1] * i % p;
8     }
9 }
10
11 int inv(int x, int p) {
12     if (x == 1) return 1;
13     return (p - p / x) * inv(p % x, p) % p;
14 }
15
16 int c(int x, int y, int p) {
17     if (x < y) return 0;
18     return fact[x] * inv(fact[y], p) % p * 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[100005];
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 }

```

```

15 int kmp(string s) {
16     int ret = 0;
17     for (int i = 0, j = 0; i < s.size(); i++) {
18         while (j && s[i] ^ t[j]) {
19             j = pmt[j - 1];
20         }
21         if (s[i] == t[j]) {
22             j++;
23         }
24         if (j == t.size()) {
25             ret++;
26             j = pmt[j - 1];
27         }
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;
3
4  int n, q;
5  int a[200005];
6  int r[200005];
7  int d[200005];
8  int cycle[200005];
9  int len[200005];
10 int cnt = 0;
11 vector<int> v[200005];
12 bitset<200005> vis1;
13 bitset<200005> vis2;
14
15 void findcycle(int x) {
16     while (!vis1[x]) {
17         vis1[x] = 1;
18         x = a[x];
19     }
20     cnt++;
21     cycle[x] = cnt;
22     r[x] = 0;
23     len[cnt] = 1;
24     int temp = a[x];
25     while (temp ^ x) {
26         r[temp] = len[cnt];
27         len[cnt]++;
28         cycle[temp] = cnt;
29         temp = a[temp];
30     }
31 }
32
33 void dfs(int x) {
34     if (vis2[x])
35         return;
36     vis2[x] = 1;
37     for (int i : v[x]) {
38         dfs(i);
39     }
40 }
41
42 void dfs2(int x) {
43     if (cycle[x] || d[x])
44         return;
45     dfs2(a[x]);
46     d[x] = d[a[x]] + 1;
47     r[x] = r[a[x]];
48     cycle[x] = cycle[a[x]];
49 }
50
51 signed main() {
52     ios::sync_with_stdio(0);
53     cin.tie(0);
54     cout.tie(0);
55     cin >> n;
56     for (int i = 1; i <= n; i++) {
57         cin >> a[i];
58         v[i].push_back(a[i]);
59         v[a[i]].push_back(i);
60     }
61     for (int i = 1; i <= n; i++) {
62         if (!vis2[i]) {
63             findcycle(i);
64             dfs(i);
65         }
66     }
67     for (int i = 1; i <= n; i++) {
68         if (!cycle[i] && !r[i]) {
69             dfs2(i);
70         }
71     }
72     for (int i = 1; i <= n; i++) {
73         cout << d[i] + len[cycle[i]] << " ";
74     }
75 }

```

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;
memset(dis, 0x3f, sizeof(dis));
11    dis[x] = 0;
pq.push({0, x});
13    while (!pq.empty()) {
        pair<int, int> now = pq.top();
15        pq.pop();
        if (vis[now.second])
17            continue;
        vis[now.second] = 1;
19        for (auto [i, w] : v[now.second]) {
            if (vis[i])
21                continue;
            if (dis[now.second] + w < dis[i]) {
23                dis[i] = dis[now.second] + w;
                pq.push({dis[i], i});
25            }
        }
27    }
}

```

4.3. MaximumFlow

```

1  #define int long long
3  using namespace std;

5  int n, m;
vector<int> v[1005];
7  int head[1005];
int c[1005][1005];
9  int lv[1005];
int ans = 0;

11 bool bfs() {
13     memset(head, 0, sizeof(head));
    memset(lv, 0, sizeof(lv));
15     queue<int> q;
    q.push(1);
17     while (!q.empty()) {
        int now = q.front();
19         q.pop();
        if (now == n)
21             continue;
        for (int i : v[now]) {
23             if (i != 1 && c[now][i] && !lv[i]) {
                lv[i] = lv[now] + 1;
25                 q.push(i);
            }
        }
27     }
    return lv[n];
29 }

31 int dfs(int x, int flow) {
33     int ret = 0;
    if (x == n)
35         return flow;
    for (int i = head[x]; i < v[x].size(); i++) {
37         int y = v[x][i];
        head[x] = y;
39         if (c[x][y] && lv[y] == lv[x] + 1) {
            int d = dfs(y, min(flow, c[x][y]));
41             flow -= d;
            c[x][y] -= d;
43             c[y][x] += d;
            ret += d;
45         }
    }
47     return ret;
}

49 signed main() {
51     cin >> n >> m;
    while (m--) {
53         int x, y, z;
        cin >> x >> y >> z;
55         if (c[x][y] || c[y][x]) {
            c[x][y] += z;
57             continue;
        }
        v[x].push_back(y);
59         v[y].push_back(x);
        c[x][y] = z;
61     }
    while (bfs()) {
63         ans += dfs(1, INT_MAX);
65     }
    cout << ans;
67 }

```

4.4. SCC

```

1  int n, m;
vector<int> v[100005];
3  int d[100005];
int low[100005];
5  int cnt = 0;
stack<int> s;
7  int scc[100005];
int now = 0;

9  void dfs(int x) {
11     d[x] = low[x] = ++cnt;
    s.push(x);
13     for (int i : v[x]) {
        if (scc[i])
15             continue;
        if (d[i]) {
17             low[x] = min(low[x], d[i]);
        } else {
19             dfs(i);
            low[x] = min(low[x], low[i]);
21         }
    }
23     if (d[x] == low[x]) {
        now++;
25         while (!s.empty()) {
            int k = s.top();
27             s.pop();
            scc[k] = now;
29             if (k == x)
                break;
31         }
    }
33 }

```

4.5. 2-SAT(CSES Giant Pizza)

```

1  #define int long long
3  using namespace std;

5  int n, m;
vector<int> v[200005];
7  int d[200005];
int low[200005];
9  int cnt = 0;
int now = 0;
11 int scc[200005];
stack<int> s;
13 int op[200005];
vector<int> v2[200005];
15 int ind[200005];
queue<int> q;
17 int ans[200005];

19 int no(int x) {
    if (x > m)
21         return x - m;
    return x + m;
23 }

25 void dfs(int x) {
    d[x] = low[x] = ++cnt;
27     s.push(x);
    for (int i : v[x]) {
29         if (scc[i])
            continue;
31         if (d[i]) {
            low[x] = min(low[x], d[i]);
33         } else {
            dfs(i);
35             low[x] = min(low[x], low[i]);
        }
    }
37     if (d[x] == low[x]) {
        now++;
39         while (!s.empty()) {
            int k = s.top();
41             s.pop();
            scc[k] = now;
43             if (k == x)
                break;
45         }
    }
47 }

49 signed main() {
51     ios::sync_with_stdio(0);
    cin.tie(0);
53     cout.tie(0);
}

```

```

cin >> n >> m;
55 while (n-->0) {
    char a, b;
57     int x, y;
    cin >> a >> x >> b >> y;
59     if (a == '-')
        x = no(x);
61     if (b == '-')
        y = no(y);
63     v[no(x)].push_back(y);
    v[no(y)].push_back(x);
65 }
    for (int i = 1; i <= 2 * m; i++) {
67         if (!d[i]) {
            dfs(i);
69         }
    }
71     for (int i = 1; i <= m; i++) {
        if (scc[i] ^ scc[i + m]) {
73             op[scc[i]] = scc[i + m];
            op[scc[i + m]] = scc[i];
75         } else {
            cout << "IMPOSSIBLE";
77             exit(0);
        }
79     }
    for (int i = 1; i <= 2 * m; i++) {
81         for (int j : v[i]) {
            if (scc[i] ^ scc[j]) {
83                 v2[scc[j]].push_back(scc[i]);
                ind[scc[i]]++;
85             }
        }
87     }
    for (int i = 1; i <= now; i++) {
89         if (!ind[i]) {
            q.push(i);
91         }
    }
93     while (!q.empty()) {
        int k = q.front();
95         q.pop();
        if (!ans[k]) {
97             ans[k] = 1;
            ans[op[k]] = 2;
99         }
        for (int i : v2[k]) {
101             ind[i]--;
            if (!ind[i]) {
103                 q.push(i);
            }
105         }
    }
107     for (int i = 1; i <= m; i++) {
        if (ans[scc[i]] == 1) {
109             cout << "+ ";
        } else {
111             cout << "- ";
        }
113     }
}

```

5. DP

5.1. Li-Chao Segment Tree

```

1 struct line {
    int a, b = 1000000000000000;
3     int y(int x) { return a * x + b; }
};

5 line tree[4000005];
7 int n, x;
int s[200005];
9 int f[200005];
int dp[200005];

11 void update(line ins, int l = 1, int r = 1e6, int index = 1) {
13     if (l == r) {
        if (ins.y(l) < tree[index].y(l)) {
15             tree[index] = ins;
        }
        return;
17     }
    int mid = (l + r) >> 1;
19     if (tree[index].a < ins.a)
        swap(tree[index], ins);
21     if (tree[index].y(mid) > ins.y(mid)) {
        swap(tree[index], ins);
23     }
}

```

```

    update(ins, l, mid, index << 1);
25 } else {
    update(ins, mid + 1, r, index << 1 | 1);
27 }
}

29 int query(int x, int l = 1, int r = 1000000, int index = 1) {
31     int cur = tree[index].y(x);
    if (l == r) {
33         return cur;
    }
35     int mid = (l + r) >> 1;
    if (x <= mid) {
37         return min(cur, query(x, l, mid, index << 1));
    } else {
39         return min(cur, query(x, mid + 1, r, index << 1 | 1));
    }
41 }

```

5.2. CHO

```

1 struct line {
    int a, b;
3     int y(int x) { return a * x + b; }
};

5 struct CHO {
    deque<line> dq;
7     int intersect(line x, line y) {
        int d1 = x.b - y.b;
9         int d2 = y.a - x.a;
        return d1 / d2;
11     }
    bool check(line x, line y, line z) {
13         int I12 = intersect(x, y);
        int I23 = intersect(y, z);
15         return I12 < I23;
    }
17     void insert(int a, int b) {
        if (!dq.empty() && a == dq.back().a)
19             return;
        while (dq.size() >= 2 &&
21             !check(dq[dq.size() - 2], dq[dq.size() - 1], {a, b})) {
            dq.pop_back();
23         }
        dq.push_back({a, b});
25     }
    void update(int x) {
27         while (dq.size() >= 2 && dq[0].y(x) >= dq[1].y(x)) {
            dq.pop_front();
29         }
    }
31     int query(int x) {
        update(x);
33         return dq.front().y(x);
    }
35 }
};

```

6. Geometry

6.1. Intersect

```

1 struct point {
    int x, y;
3     point operator+(point b) { return {x + b.x, y + b.y}; }
    point operator-(point b) { return {x - b.x, y - b.y}; }
5     int operator*(point b) { return x * b.x + y * b.y; }
    int operator^(point b) { return x * b.y - y * b.x; }
7 };

9 bool onseg(point x, point y, point z) {
    return ((x - z) ^ (y - z)) == 0 && (x - z) * (y - z) <= 0;
11 }

13 int dir(point x, point y) {
    int k = x ^ y;
15     if (k == 0)
        return 0;
    if (k > 0)
17         return 1;
    return -1;
19 }

21 bool intersect(point x, point y, point z, point w) {
23     if (onseg(x, y, z) || onseg(x, y, w))
        return 1;
    if (onseg(z, w, x) || onseg(z, w, y))
25         return 1;
    if (dir(y - x, z - x) * dir(y - x, w - x) == -1 &&
27     )
        return 1;
    return 0;
}

```

```

    dir(z - w, x - w) * dir(z - w, y - w) == -1) {
29     return 1;
    }
31     return 0;
    }

```

6.2. Inside

```

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

```

6.3. Minimum Euclidean Distance

```

1 #define int long long
3 #define pii pair<int, int>
    using namespace std;
5
    int n;
7    vector<pair<int, int>> v;
    set<pair<int, int>> s;
9    int dd = LONG_LONG_MAX;
11
12 int dis(pii x, pii y) {
    return (x.first - y.first) * (x.first - y.first) +
13         (x.second - y.second) * (x.second - y.second);
    }
15
16 signed main() {
17     ios::sync_with_stdio(0);
    cin.tie(0);
19     cout.tie(0);
    cin >> n;
21     for (int i = 0; i < n; i++) {
        int x, y;
23         cin >> x >> y;
        x += 1000000000;
25         v.push_back({x, y});
    }
27     sort(v.begin(), v.end());
    int l = 0;
29     for (int i = 0; i < n; i++) {
        int d = ceil(sqrt(dd));
31         while (l < i && v[l].first - v[i].first > d) {
            s.erase({v[l].second, v[l].first});
33             l++;
        }
35         auto x = s.lower_bound({v[i].second - d, 0});
        auto y = s.upper_bound({v[i].second + d, 0});
37         for (auto it = x; it != y; it++) {
            dd = min(dd, dis({it->second, it->first}, v[i]));
39         }
        s.insert({v[i].second, v[i].first});
41     }
    cout << dd;
43 }

```

7. Tree

7.1. Heavy Light Decomposition (modify and query on path)

```

1 #define int long long
3 using namespace std;
5
    int tree[800005];
7
    int n, q;
    int a[200005];
9    int st[200005];
    int tp[200005];
11    int p[200005];
    int cnt = 0;
13    int d[200005];

```

```

    int si[200005];
15    vector<int> v[200005];
    int b[200005];
17
    void build(int l = 1, int r = n, int index = 1) {
19         if (l == r) {
            tree[index] = b[l];
21             return;
        }
        int mid = (l + r) >> 1;
23         build(l, mid, index << 1);
        build(mid + 1, r, index << 1 | 1);
25         tree[index] = max(tree[index << 1], tree[index << 1 | 1]);
27     }
29
    int query(int L, int R, int l = 1, int r = n, int index = 1) {
        if (L == l && r == R) {
31             return tree[index];
        }
        int mid = (l + r) >> 1;
33         if (R <= mid) {
            return query(L, R, l, mid, index << 1);
35         }
        if (L > mid) {
            return query(L, R, mid + 1, r, index << 1 | 1);
37         }
        return max(query(L, mid, l, mid, index << 1),
39                  query(mid + 1, R, mid + 1, r, index << 1 | 1));
41     }
43
    void modify(int x, int val, int l = 1, int r = n, int index = 1) {
45         if (l == r) {
            tree[index] = val;
47             return;
        }
        int mid = (l + r) >> 1;
49         if (x <= mid) {
            modify(x, val, l, mid, index << 1);
51         } else {
            modify(x, val, mid + 1, r, index << 1 | 1);
53         }
        tree[index] = max(tree[index << 1], tree[index << 1 | 1]);
55     }
57
    void dfs(int x, int pre) {
59         si[x] = 1;
        for (int i : v[x]) {
61             if (i == pre)
                continue;
63             p[i] = x;
            d[i] = d[x] + 1;
65             dfs(i, x);
            si[x] += si[i];
67         }
    }
69
    void dfs2(int x, int pre, int t) {
71         tp[x] = t;
        st[x] = ++cnt;
73         int ma = 0;
        for (int i : v[x]) {
75             if (i == pre)
                continue;
77             if (si[i] > si[ma]) {
                ma = i;
79             }
        }
81         if (!ma)
            return;
            dfs2(ma, x, t);
83         for (int i : v[x]) {
            if (i == pre || i == ma) {
85                 continue;
            }
            dfs2(i, x, i);
87         }
    }
89
    int f(int x, int y) {
        int ret = 0;
93         while (tp[x] ^ tp[y]) {
            while (d[tp[x]] < d[tp[y]]) {
95                 swap(x, y);
            }
            ret = max(ret, query(st[tp[x]], st[x]));
97             x = p[tp[x]];
        }
        if (d[x] > d[y])
101         swap(x, y);
        ret = max(ret, query(st[x], st[y]));
103     }

```

```

    return ret;
105 }

107 signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin >> n >> q;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    for (int i = 1; i < n; i++) {
        int x, y;
        cin >> x >> y;
        v[x].push_back(y);
        v[y].push_back(x);
    }
    dfs(1, 0);
    dfs2(1, 0, 1);
    for (int i = 1; i <= n; i++) {
        b[st[i]] = a[i];
    }
    build();
    while (q--) {
        int mode, x, y;
        cin >> mode >> x >> y;
        if (mode == 1) {
            modify(st[x], y);
        } else {
            cout << f(x, y) << " ";
        }
    }
135 }
}

```

```

55     int x, y;
        cin >> x >> y;
        v[x].push_back(y);
        v[y].push_back(x);
59     }
    dfs(1, 0);
    init();
    while (q--) {
        int x, y;
        cin >> x >> y;
        int k = lca(x, y);
        cout << (d[x] + d[y] - 2 * d[k]) << "\n";
67     }
}

```

7.2. LCA

```

1  #define int long long
3  using namespace std;

5  int n, q;
   int a[200005][21];
   int d[200005];
   vector<int> v[200005];

9  void init() {
11     for (int j = 1; j < 21; j++) {
        for (int i = 1; i <= n; i++) {
            a[i][j] = a[a[i][j - 1]][j - 1];
        }
15     }
}

17 void dfs(int x, int pre) {
19     for (int i : v[x]) {
        if (i == pre) {
            continue;
        }
        a[i][0] = x;
        d[i] = d[x] + 1;
        dfs(i, x);
    }
27 }

29 int lca(int x, int y) {
    while (d[x] ^ d[y]) {
        if (d[x] < d[y]) {
            swap(x, y);
        }
        int k = __lg(d[x] - d[y]);
        x = a[x][k];
    }
    if (x == y) {
        return x;
    }
    for (int i = 20; i >= 0; i--) {
        if (a[x][i] != a[y][i]) {
            x = a[x][i];
            y = a[y][i];
        }
    }
    return a[x][0];
47 }

49 signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin >> n >> q;
    for (int i = 1; i < n; i++) {

```