

Contents

1 DataStructure

- 1.1 Treap
- 1.2 Dynamic Segment Tree

2 Math

- 2.1 FFT
- 2.2 NTT
- 2.3 Gaussian-Jordan
- 2.4 Mu
- 2.5 Lucas
- 2.6 Inv
- 2.7 Formula
 - 2.7.1 Dirichlet Convolution
 - 2.7.2 Burnside's Lemma
 - 2.7.3 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 + treap[rson].sum + treap[index].val;
20 }
21
22 void push(int index) {
23     if (!treap[index].tag)
24         return;
25     swap(treap[index].lson, treap[index].rson);
26     int lson = treap[index].lson;
27     int rson = treap[index].rson;
28     treap[lson].tag ^= 1;
29     treap[rson].tag ^= 1;
30     treap[index].tag = 0;
31 }
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     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;

```

```

29 }
31 void modify(int l, int r, int L, int R, int val, int index) {
32     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 }
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. FFT

```

1 using namespace std;
2 inline int read() {
3     int ans = 0;
4     char c = getchar();
5     while (!isdigit(c))
6         c = getchar();
7     while (isdigit(c)) {
8         ans = ans * 10 + c - '0';
9         c = getchar();
10    }
11    return ans;
12 }
13 typedef complex<double> comp;
14 const int MAXN = 1000005;
15 const comp I(0, 1);
16 const double PI = acos(-1);
17 comp A[MAXN * 3], B[MAXN * 3], tmp[MAXN * 3], ans[MAXN * 3];
18 void fft(comp F[], int N, int sgn = 1) {
19     if (N == 1)

```

```

21     return;
22     memcp(tmp, F, sizeof(comp) * N);
23     for (int i = 0; i < N; i++)
24         *(i % 2 ? F + i / 2 + N / 2 : F + i / 2) = tmp[i];
25     fft(F, N / 2, sgn), fft(F + N / 2, N / 2, sgn);
26     comp *G = F, *H = F + N / 2;
27     comp cur = 1, step = exp(2 * PI / N * sgn * I);
28     for (int k = 0; k < N / 2; k++) {
29         tmp[k] = G[k] + cur * H[k];
30         tmp[k + N / 2] = G[k] - cur * H[k];
31         cur *= step;
32     }
33     memcp(F, tmp, sizeof(comp) * N);
34 }
35 int main() {
36     int n = read(), m = read(), N = 1 << __lg(n + m + 1) + 1;
37     for (int i = 0; i <= n; ++i)
38         A[i] = read();
39     for (int i = 0; i <= m; ++i)
40         B[i] = read();
41     fft(A, N), fft(B, N);
42     for (int i = 0; i < N; ++i)
43         ans[i] = A[i] * B[i];
44     fft(ans, N, -1);
45     for (int i = 0; i <= n + m; ++i)
46         printf("%d ", int(ans[i].real() / N + 0.1));
47     return 0;
48 }

```

2.2. NTT

```

1 #define ll long long
2 using namespace std;
3
4 const int MAXN = 1000005;
5 const int MOD = 998244353, G = 3;
6 int rev[MAXN * 3];
7
8 int qpow(int x, int y) {
9     int ret = 1;
10    while (y) {
11        if (y & 1) {
12            ret *= x;
13            ret %= MOD;
14        }
15        x *= x;
16        x %= MOD;
17        y >>= 1;
18    }
19    return ret;
20 }
21
22 void ntt(int F[], int N, int sgn) {
23     int bit = __lg(N);
24     for (int i = 0; i < N; ++i) {
25         rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
26         if (i < rev[i])
27             swap(F[i], F[rev[i]]);
28     }
29     for (int l = 1, t = 1; l < N; l <= 1, t++) {
30         int step = qpow(G, ((MOD - 1) >> t) * sgn + MOD - 1);
31         for (int i = 0; i < N; i += l << 1)
32             for (int k = i, cur = 1; k < i + l; ++k) {
33                 int g = F[k], h = (ll)F[k + l] * cur % MOD;
34                 F[k] = (g + h) % MOD;
35                 F[k + l] = ((g - h) % MOD + MOD) % MOD;
36                 cur = (ll)cur * step % MOD;
37             }
38     }
39     if (sgn == -1) {
40         int invN = qpow(N, MOD - 2);
41         for (int i = 0; i < N; ++i)
42             F[i] = (ll)F[i] * invN % MOD;
43     }
44 }

```

2.3. Gaussian-Jordan

```

1 #define int long long
2 using namespace std;
3
4 int n;
5 double a[105][105];
6
7 // n <= m
8 void gaussian(double a[105][105], int n, int m) {
9     int curi = 0;
10    for (int j = 0; j < m; j++) {

```

```

13     int i;
14     for (i = curi; i < n; i++) {
15         if (a[i][j]) {
16             break;
17         }
18     }
19     if (a[i][j] == 0)
20         continue;
21     for (int k = 0; k < m; k++) {
22         swap(a[i][k], a[curi][k]);
23     }
24     for (int k = m - 1; k >= j; k--) {
25         a[curi][k] /= a[curi][j];
26     }
27     for (int i = 0; i < n; ++i) {
28         if (i != curi) {
29             for (int k = m - 1; k >= j; k--) {
30                 a[i][k] -= a[curi][k] * a[i][j];
31             }
32         }
33     }
34     curi++;
35 }

```

2.4. 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)
14                break;
15            vis[i * p] = 1;
16            if (i % p == 0) {
17                mu[i * p] = 0;
18                break;
19            } else {
20                mu[i * p] = mu[i] * mu[p];
21            }
22        }
23    }
24 }

```

2.5. Lucas

```

1 int fact[1000005];
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)
13         return 1;
14     return (p - p / x) * inv(p % x, p) % p;
15 }
16
17 int c(int x, int y, int p) {
18     if (x < y)
19         return 0;
20     int k = fact[x] * inv(fact[y], p) % p;
21     return k * inv(fact[x - y], p) % p;
22 }
23
24 int lucas(int x, int y, int p) {
25     if (x == 0)
26         return 1;
27     return lucas(x / p, y / p, p) % p * c(x % p, y % p, p) % p;
28 }

```

2.6. 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 }

```

2.7. Formula

2.7.1. Dirichlet Convolution

$$\varepsilon = \mu * 1$$

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

2.7.2. 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

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

2.7.3. 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])
10             j++;
11         pmt[i] = j;
12     }
13 }
14
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) {
15             s += '0';
16         } else {
17             s += t[i / 2];

```

```

19 }
20 }
21 n = s.size();
22 d[0] = 1;
23 for (int i = 0, l = 0, r = 0; i < n; i++) {
24     if (i > r) {
25         d[i] = 1;
26         bool a = i + d[i] < n;
27         bool b = i - d[i] >= 0;
28         bool c = (s[i + d[i]] == s[i - d[i]]);
29         while (a && b && c) {
30             d[i]++;
31             a = i + d[i] < n;
32             b = i - d[i] >= 0;
33             c = (s[i + d[i]] == s[i - d[i]]);
34         }
35         l = i - d[i] + 1;
36         r = i + d[i] - 1;
37     } else {
38         int j = l + r - i;
39         if (j - d[j] + 1 > l) {
40             d[i] = d[j];
41         } else {
42             d[i] = r - i + 1;
43             a = i + d[i] < n;
44             b = i - d[i] >= 0;
45             c = (s[i + d[i]] == s[i - d[i]]);
46             while (a && b && c) {
47                 d[i]++;
48                 a = i + d[i] < n;
49                 b = i - d[i] >= 0;
50                 c = (s[i + d[i]] == s[i - d[i]]);
51             }
52             l = i - d[i] + 1;
53             r = i + d[i] - 1;
54         }
55     }
56     // cout << d[i] << " ";
57     if (d[i] > d[ans]) {
58         ans = i;
59     }
60 }
61 for (int i = ans - d[ans] + 1; i < ans + d[ans]; i++) {
62     if (s[i] ^ '0') {
63         cout << s[i];
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;

```

```

39     for (int i : v[x]) {
40         dfs(i);
41     }
42 }
43
44 void dfs2(int x) {
45     if (cycle[x] || d[x])
46         return;
47     dfs2(a[x]);
48     d[x] = d[a[x]] + 1;
49     r[x] = r[a[x]];
50     cycle[x] = cycle[a[x]];
51 }
52
53 signed main() {
54     ios::sync_with_stdio(0);
55     cin.tie(0);
56     cout.tie(0);
57     cin >> n;
58     for (int i = 1; i <= n; i++) {
59         cin >> a[i];
60         v[i].push_back(a[i]);
61         v[a[i]].push_back(i);
62     }
63     for (int i = 1; i <= n; i++) {
64         if (!vis2[i]) {
65             findcycle(i);
66             dfs(i);
67         }
68     }
69     for (int i = 1; i <= n; i++) {
70         if (!cycle[i] && !r[i]) {
71             dfs2(i);
72         }
73     }
74     for (int i = 1; i <= n; i++) {
75         cout << d[i] + len[cycle[i]] << " ";
76     }
77 }

```

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

```

```

    int now = q.front();
    q.pop();
    if (now == n)
        continue;
    for (int i : v[now]) {
        if (i != 1 && c[now][i] && !lv[i]) {
            lv[i] = lv[now] + 1;
            q.push(i);
        }
    }
}
return lv[n];
}

int dfs(int x, int flow) {
    int ret = 0;
    if (x == n)
        return flow;
    for (int i = head[x]; i < v[x].size(); i++) {
        int y = v[x][i];
        head[x] = y;
        if (c[x][y] && lv[y] == lv[x] + 1) {
            int d = dfs(y, min(flow, c[x][y]));
            flow -= d;
            c[x][y] -= d;
            c[y][x] += d;
            ret += d;
        }
    }
    return ret;
}

signed main() {
    cin >> n >> m;
    while (m--) {
        int x, y, z;
        cin >> x >> y >> z;
        if (c[x][y] || c[y][x]) {
            c[x][y] += z;
            continue;
        }
        v[x].push_back(y);
        v[y].push_back(x);
        c[x][y] = z;
    }
    while (bfs()) {
        ans += dfs(1, INT_MAX);
    }
    cout << ans;
}

```

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

```

```

3 using namespace std;
4
5 int n, m;
6 vector<int> v[200005];
7 int d[200005];
8 int low[200005];
9 int cnt = 0;
10 int now = 0;
11 int scc[200005];
12 stack<int> s;
13 int op[200005];
14 vector<int> v2[200005];
15 int ind[200005];
16 queue<int> q;
17 int ans[200005];
18
19 int no(int x) {
20     if (x > m)
21         return x - m;
22     return x + m;
23 }
24
25 void dfs(int x) {
26     d[x] = low[x] = ++cnt;
27     s.push(x);
28     for (int i : v[x]) {
29         if (scc[i])
30             continue;
31         if (d[i]) {
32             low[x] = min(low[x], d[i]);
33         } else {
34             dfs(i);
35             low[x] = min(low[x], low[i]);
36         }
37     }
38     if (d[x] == low[x]) {
39         now++;
40         while (!s.empty()) {
41             int k = s.top();
42             s.pop();
43             scc[k] = now;
44             if (k == x)
45                 break;
46         }
47     }
48 }
49
50 signed main() {
51     ios::sync_with_stdio(0);
52     cin.tie(0);
53     cout.tie(0);
54     cin >> n >> m;
55     while (n--) {
56         char a, b;
57         int x, y;
58         cin >> a >> x >> b >> y;
59         if (a == '-')
60             x = no(x);
61         if (b == '-')
62             y = no(y);
63         v[no(x)].push_back(y);
64         v[no(y)].push_back(x);
65     }
66     for (int i = 1; i <= 2 * m; i++) {
67         if (!d[i]) {
68             dfs(i);
69         }
70     }
71     for (int i = 1; i <= m; i++) {
72         if (scc[i] ^ scc[i + m]) {
73             op[scc[i]] = scc[i + m];
74             op[scc[i + m]] = scc[i];
75         } else {
76             cout << "IMPOSSIBLE";
77             exit(0);
78         }
79     }
80     for (int i = 1; i <= 2 * m; i++) {
81         for (int j : v[i]) {
82             if (scc[i] ^ scc[j]) {
83                 v2[scc[j]].push_back(scc[i]);
84                 ind[scc[i]]++;
85             }
86         }
87     }
88     for (int i = 1; i <= now; i++) {
89         if (!ind[i]) {
90             q.push(i);
91         }
92     }
93 }

```

```

93 while (!q.empty()) {
94     int k = q.front();
95     q.pop();
96     if (!ans[k]) {
97         ans[k] = 1;
98         ans[op[k]] = 2;
99     }
100     for (int i : v2[k]) {
101         ind[i]--;
102         if (!ind[i]) {
103             q.push(i);
104         }
105     }
106 }
107 for (int i = 1; i <= m; i++) {
108     if (ans[scc[i]] == 1) {
109         cout << "+ ";
110     } else {
111         cout << "- ";
112     }
113 }

```

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
30 int query(int x, int l = 1, int r = 1000000, int index = 1) {
31     int cur = tree[index].y(x);
32     if (l == r) {
33         return cur;
34     }
35     int mid = (l + r) >> 1;
36     if (x <= mid) {
37         return min(cur, query(x, l, mid, index << 1));
38     } else {
39         return min(cur, query(x, mid + 1, r, index << 1 | 1));
40     }
41 }

```

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<pii> v;
7 set<pii> 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) +

```

```

13         (x.second - y.second) * (x.second - y.second);
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[l].first - v[l].second > 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     }
43 }

```

7. Tree

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

```

1  #define int long long
2  using namespace std;
3
4  int tree[800005];
5
6  int n, q;
7  int a[200005];
8  int st[200005];
9  int tp[200005];
10 int p[200005];
11 int cnt = 0;
12 int d[200005];
13 int si[200005];
14 vector<int> v[200005];
15 int b[200005];
16
17 void build(int l = 1, int r = n, int index = 1) {
18     if (l == r) {
19         tree[index] = b[l];
20         return;
21     }
22     int mid = (l + r) >> 1;
23     build(l, mid, index << 1);
24     build(mid + 1, r, index << 1 | 1);
25     tree[index] = max(tree[index << 1], tree[index << 1 | 1]);
26 }
27
28 int query(int L, int R, int l = 1, int r = n, int index = 1) {
29     if (L == l && R == r) {
30         return tree[index];
31     }
32     int mid = (l + r) >> 1;
33     if (R <= mid) {
34         return query(L, R, l, mid, index << 1);
35     }
36     if (L > mid) {
37         return query(L, R, mid + 1, r, index << 1 | 1);
38     }
39     return max(query(L, mid, l, mid, index << 1),
40               query(mid + 1, R, mid + 1, r, index << 1 | 1));
41 }
42
43 void modify(int x, int val, int l = 1, int r = n, int index = 1) {
44     if (l == r) {
45         tree[index] = val;
46         return;
47     }
48     int mid = (l + r) >> 1;
49     if (x <= mid) {
50         modify(x, val, l, mid, index << 1);
51     } else {

```

```

52         modify(x, val, mid + 1, r, index << 1 | 1);
53     }
54     tree[index] = max(tree[index << 1], tree[index << 1 | 1]);
55 }
56
57 void dfs(int x, int pre) {
58     si[x] = 1;
59     for (int i : v[x]) {
60         if (i == pre)
61             continue;
62         p[i] = x;
63         d[i] = d[x] + 1;
64         dfs(i, x);
65         si[x] += si[i];
66     }
67 }
68
69 void dfs2(int x, int pre, int t) {
70     tp[x] = t;
71     st[x] = ++cnt;
72     int ma = 0;
73     for (int i : v[x]) {
74         if (i == pre)
75             continue;
76         if (si[i] > si[ma]) {
77             ma = i;
78         }
79     }
80     if (!ma)
81         return;
82     dfs2(ma, x, t);
83     for (int i : v[x]) {
84         if (i == pre || i == ma) {
85             continue;
86         }
87         dfs2(i, x, i);
88     }
89 }
90
91 int f(int x, int y) {
92     int ret = 0;
93     while (tp[x] ^ tp[y]) {
94         if (d[tp[x]] < d[tp[y]]) {
95             swap(x, y);
96         }
97         ret = max(ret, query(st[tp[x]], st[x]));
98         x = p[tp[x]];
99     }
100     if (d[x] > d[y])
101         swap(x, y);
102     ret = max(ret, query(st[x], st[y]));
103     return ret;
104 }
105
106 signed main() {
107     ios::sync_with_stdio(0);
108     cin.tie(0);
109     cout.tie(0);
110     cin >> n >> q;
111     for (int i = 1; i <= n; i++) {
112         cin >> a[i];
113     }
114     for (int i = 1; i < n; i++) {
115         int x, y;
116         cin >> x >> y;
117         v[x].push_back(y);
118         v[y].push_back(x);
119     }
120     dfs(1, 0);
121     dfs2(1, 0, 1);
122     for (int i = 1; i <= n; i++) {
123         b[st[i]] = a[i];
124     }
125     build();
126     while (q--) {
127         int mode, x, y;
128         cin >> mode >> x >> y;
129         if (mode == 1) {
130             modify(st[x], y);
131         } else {
132             cout << f(x, y) << " ";
133         }
134     }
135 }

```

7.2. LCA

```

1  #define int long long
2  using namespace std;
3

```



```

5  int n, q;
   int a[200005][21];
7  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++) {
13         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) {
21         continue;
         }
23         a[i][0] = x;
         d[i] = d[x] + 1;
25         dfs(i, x);
     }
27 }

29 int lca(int x, int y) {
   while (d[x] ^ d[y]) {
31     if (d[x] < d[y]) {
         swap(x, y);
     }
33     int k = _lg(d[x] - d[y]);
35     x = a[x][k];
   }
37   if (x == y) {
       return x;
39   }
   for (int i = 20; i >= 0; i--) {
41     if (a[x][i] != a[y][i]) {
         x = a[x][i];
43         y = a[y][i];
     }
45   }
   return a[x][0];
47 }

49 signed main() {
   ios::sync_with_stdio(0);
51   cin.tie(0);
   cout.tie(0);
53   cin >> n >> q;
   for (int i = 1; i < n; i++) {
55     int x, y;
     cin >> x >> y;
57     v[x].push_back(y);
     v[y].push_back(x);
59   }
   dfs(1, 0);
61   init();
   while (q--) {
63     int x, y;
     cin >> x >> y;
65     int k = lca(x, y);
     cout << (d[x] + d[y] - 2 * d[k]) << "\n";
67   }
}

```