

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
- 2.8 Matrix

3 String

- 3.1 KMP
- 3.2 Longest Palindrome
- 3.3 Z

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

8 Misc

- 8.1 Tri Search

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 }

```

```

35 pii split(int rk, int index) {
36     if (!index)
37         return {0, 0};
38     push(index);
39     int lson = treap[index].lson;
40     int rson = treap[index].rson;
41     if (rk <= treap[lson].si) {
42         pii temp = split(rk, lson);
43         treap[index].lson = temp.second;
44         update(index);
45         return {temp.first, index};
46     } else {
47         pii temp = split(rk - treap[lson].si - 1, rson);
48         treap[index].rson = temp.first;
49         update(index);
50         return {index, temp.second};
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 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 int query(int l, int r) {
81     pii R = split(r, root);
82     pii L = split(l - 1, R.first);
83     int ret = treap[L.second].sum;
84     R.first = merge(L.first, L.second);
85     root = merge(R.first, R.second);
86     return ret;
87 }
88
89 void modify(int l, int r) {
90     pii R = split(r, root);
91     pii L = split(l - 1, R.first);
92     treap[L.second].tag ^= 1;
93     R.first = merge(L.first, L.second);
94     root = merge(R.first, R.second);
95 }

```

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 int lson = tree[index].lson;

```

```

    int rson = tree[index].rson;
25 tree[lson].tag += tree[index].tag;
    tree[rson].tag += tree[index].tag;
27 tree[index].data = tree[index].rv();
    tree[index].tag = 0;
29 }

31 void modify(int l, int r, int L, int R, int val, int index) {
    if (l == L && r == R) {
33         tree[index].tag += val;
        return;
35     }
    int mid = (l + r) >> 1;
    push(index);
    int lson = tree[index].lson;
    int rson = tree[index].rson;
    if (R <= mid) {
41         modify(l, mid, L, R, val, lson);
    } else if (L > mid) {
43         modify(mid + 1, r, L, R, val, rson);
    } else {
45         modify(l, mid, L, mid, val, lson);
        modify(mid + 1, r, mid + 1, R, val, rson);
47     }
    tree[index].data = tree[lson].rv() + tree[rson].rv();
49 }

51 int query(int l, int r, int L, int R, int index) {
    // cout << L << " " << R << "\n";
    if (l == L && r == R) {
53         return tree[index].rv();
    }
    int mid = (l + r) >> 1;
    push(index);
    int lson = tree[index].lson;
    int rson = tree[index].rson;
    if (R <= mid) {
61         return query(l, mid, L, R, lson);
    }
    if (L > mid) {
63         return query(mid + 1, r, L, R, rson);
    }
    return query(l, mid, L, mid, lson) + query(mid + 1, r, mid + 1, R, rson);
65 }

67 }

69 signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin >> n >> q;
    for (int i = 1; i <= n; i++) {
75         cin >> a[i];
        modify(1, mx, a[i], a[i], 1, 1);
77     }
    while (q--) {
79         char mode;
        int x, y;
        cin >> mode;
        if (mode == '?') {
83             cin >> x >> y;
            cout << query(1, mx, x, y, 1) << "\n";
85         } else {
            cin >> x >> y;
            modify(1, mx, a[x], a[x], -1, 1);
            a[x] = y;
            modify(1, mx, a[x], a[x], 1, 1);
89         }
91     }
}

```

2. Math

2.1. FFT

```

1 using namespace std;
3 inline int read() {
    int ans = 0;
    char c = getchar();
    while (!isdigit(c))
        c = getchar();
    while (isdigit(c)) {
9         ans = ans * 10 + c - '0';
        c = getchar();
11     }
    return ans;
13 }
typedef complex<double> comp;
15 const int MAXN = 1000005;

```

```

const comp I(0, 1);
17 const double PI = acos(-1);
comp A[MAXN * 3], B[MAXN * 3], tmp[MAXN * 3], ans[MAXN * 3];
19 void fft(comp F[], int N, int sgn = 1) {
    if (N == 1)
        return;
    memcpy(tmp, F, sizeof(comp) * N);
    for (int i = 0; i < N; i++)
        *(i % 2 ? F + i / 2 + N / 2 : F + i / 2) = tmp[i];
    fft(F, N / 2, sgn), fft(F + N / 2, N / 2, sgn);
    comp *G = F, *H = F + N / 2;
    comp cur = 1, step = exp(2 * PI / N * sgn * I);
    for (int k = 0; k < N / 2; k++) {
29         tmp[k] = G[k] + cur * H[k];
        tmp[k + N / 2] = G[k] - cur * H[k];
        cur *= step;
    }
    memcpy(F, tmp, sizeof(comp) * N);
33 }

35 int main() {
    int n = read(), m = read(), N = 1 << __lg(n + m + 1) + 1;
    for (int i = 0; i <= n; ++i)
        A[i] = read();
    for (int i = 0; i <= m; ++i)
        B[i] = read();
    fft(A, N), fft(B, N);
    for (int i = 0; i < N; ++i)
        ans[i] = A[i] * B[i];
    fft(ans, N, -1);
    for (int i = 0; i <= n + m; ++i)
        printf("%d ", int(ans[i].real() / N + 0.1));
47     return 0;
}

```

2.2. NTT

```

1 #define ll long long
3 using namespace std;

5 const int MAXN = 1000005;
const int MOD = 998244353, G = 3;
int rev[MAXN * 3];

9 int qpow(int x, int y) {
    int ret = 1;
    while (y) {
        if (y & 1)
            ret *= x;
        x *= x;
        x %= MOD;
        y >>= 1;
    }
    return ret;
21 }

23 void ntt(int F[], int N, int sgn) {
    int bit = __lg(N);
    for (int i = 0; i < N; ++i) {
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
        if (i < rev[i])
            swap(F[i], F[rev[i]]);
    }
    for (int l = 1, t = 1; l < N; l <= 1, t++) {
        int step = qpow(G, ((MOD - 1) >> t) * sgn + MOD - 1);
        for (int i = 0; i < N; i += l << 1)
            for (int k = i, cur = 1; k < i + l; ++k) {
                int g = F[k], h = (ll)F[k + l] * cur % MOD;
                F[k] = (g + h) % MOD;
                F[k + l] = ((g - h) % MOD + MOD) % MOD;
                cur = (ll)cur * step % MOD;
            }
    }
    if (sgn == -1) {
        int invN = qpow(N, MOD - 2);
        for (int i = 0; i < N; ++i)
            F[i] = (ll)F[i] * invN % MOD;
45 }
}

```

2.3. Gaussian-Jordan

```

1 #define int long long
3 using namespace std;

5 int n;
double a[105][105];

```

```

7 // n <= m
9 void gaussian(double a[105][105], int n, int m) {
11     int curi = 0;
12     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             if (a[i][j] == 0)
19                 continue;
20             for (int k = 0; k < m; k++) {
21                 swap(a[i][k], a[curi][k]);
22             }
23             for (int k = m - 1; k >= j; k--) {
24                 a[curi][k] /= a[curi][j];
25             }
26             for (int i = 0; i < n; ++i) {
27                 if (i != curi) {
28                     for (int k = m - 1; k >= j; k--) {
29                         a[i][k] -= a[curi][k] * a[i][j];
30                     }
31                 }
32             }
33             curi++;
34         }
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    }
}

```

2.5. 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)
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

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

2.8. Matrix

```

1 #define int long long
2 using namespace std;
3
4 template <class T> T extgcd(T a, T b, T &x, T &y) {
5     if (!b) {
6         x = 1;
7         y = 0;
8         return a;
9     }
10    T ans = extgcd(b, a % b, y, x);
11    y -= a / b * x;
12    return ans;
13 }
14
15 template <class T> T modeq(T a, T b, T p) {
16    T x, y, d = extgcd(a, p, x, y);
17    if (b % d)
18        return 0;
19    return ((b / d * x) % p + p) % p;
20 }
21
22 template <class T> class Matrix {
23     static const T MOD = 1000000007;
24
25 public:
26     vector<vector<T>> v;
27     Matrix(int n, int m, int identity) {
28         v = vector<vector<T>>(n, vector<T>(m, 0));
29         if (identity)
30             for (int i = 0, k = min(n, m); i < k; ++i)
31                 v[i][i] = 1;
32     }
33     Matrix(Matrix &b) { v = b.v; }
34     void in(int l = 0, int m = -1, int u = 0, int n = -1) {
35         if (n < 0)
36             n = v.size();
37         if (m < 0)
38             m = v[0].size();
39         for (int i = u; i < n; ++i)
40             for (int j = l; j < m; ++j)
41                 scanf("%lld", &v[i][j]);
42     }
43     Matrix(int n, int m) {
44         v = vector<vector<T>>(n, vector<T>(m, 0));
45         in();
46     }
47 }

```

```

void out(int l = 0, int m = -1, int u = 0, int n = -1) {
    if (n < 0)
        n = v.size();
    if (m < 0)
        m = v[0].size();
    for (int i = u; i < n; ++i)
        for (int j = l; j < m; ++j)
            printf("%lld%c", v[i][j], " \n"[j == m - 1]);
}

Matrix operator=(Matrix &b) {
    v = b.v;
    return *this;
}

Matrix operator+(Matrix &b) {
    Matrix ans(*this);
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j) {
            ans.v[i][j] += b.v[i][j];
            if (MOD) {
                if (ans.v[i][j] < 0)
                    ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
                if (ans.v[i][j] >= MOD)
                    ans.v[i][j] %= MOD;
            }
        }
    return ans;
}

Matrix operator+(T x) {
    Matrix ans(*this);
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j) {
            ans.v[i][j] += x;
            if (MOD) {
                if (ans.v[i][j] < 0)
                    ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
                if (ans.v[i][j] >= MOD)
                    ans.v[i][j] %= MOD;
            }
        }
    return ans;
}

Matrix operator-(Matrix &b) {
    Matrix ans(*this);
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j) {
            ans.v[i][j] -= b.v[i][j];
            if (MOD) {
                if (ans.v[i][j] < 0)
                    ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
                if (ans.v[i][j] >= MOD)
                    ans.v[i][j] %= MOD;
            }
        }
    return ans;
}

Matrix operator-(T x) {
    Matrix ans(*this);
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j) {
            ans.v[i][j] -= x;
            if (MOD) {
                if (ans.v[i][j] < 0)
                    ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
                if (ans.v[i][j] >= MOD)
                    ans.v[i][j] %= MOD;
            }
        }
    return ans;
}

Matrix operator+=(Matrix &b) {
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j) {
            v[i][j] += b.v[i][j];
            if (MOD) {
                if (v[i][j] < 0)
                    v[i][j] = (v[i][j] % MOD + MOD) % MOD;
                if (v[i][j] >= MOD)
                    v[i][j] %= MOD;
            }
        }
    return *this;
}

Matrix operator+=(T x) {
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)

```

```

        for (int j = 0; j < m; ++j) {
            v[i][j] += x;
            if (MOD) {
                if (v[i][j] < 0)
                    v[i][j] = (v[i][j] % MOD + MOD) % MOD;
                if (v[i][j] >= MOD)
                    v[i][j] %= MOD;
            }
        }
    }
    return *this;
}

Matrix operator-=(Matrix &b) {
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j) {
            v[i][j] -= b.v[i][j];
            if (MOD) {
                if (v[i][j] < 0)
                    v[i][j] = (v[i][j] % MOD + MOD) % MOD;
                if (v[i][j] >= MOD)
                    v[i][j] %= MOD;
            }
        }
    return *this;
}

Matrix operator-=(T x) {
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j) {
            v[i][j] -= x;
            if (MOD) {
                if (v[i][j] < 0)
                    v[i][j] = (v[i][j] % MOD + MOD) % MOD;
                if (v[i][j] >= MOD)
                    v[i][j] %= MOD;
            }
        }
    return *this;
}

Matrix operator*(Matrix &b) {
    int n = v.size();
    int p = b.v.size();
    int m = b.v[0].size();
    Matrix ans(n, m, 0);
    for (int i = 0; i < n; ++i)
        for (int k = 0; k < p; ++k)
            for (int j = 0; j < m; ++j) {
                ans.v[i][j] += v[i][k] * b.v[k][j];
                if (MOD) {
                    if (ans.v[i][j] < 0)
                        ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
                    if (ans.v[i][j] >= MOD)
                        ans.v[i][j] %= MOD;
                }
            }
    return ans;
}

Matrix operator*(T x) {
    Matrix ans(*this);
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j) {
            ans.v[i][j] *= x;
            if (MOD) {
                if (ans.v[i][j] < 0)
                    ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
                if (ans.v[i][j] >= MOD)
                    ans.v[i][j] %= MOD;
            }
        }
    return ans;
}

Matrix operator*=(Matrix &b) {
    int n = v.size();
    int p = b.v.size();
    int m = b.v[0].size();
    Matrix ans(n, m, 0);
    for (int i = 0; i < n; ++i)
        for (int k = 0; k < p; ++k)
            for (int j = 0; j < m; ++j) {
                ans.v[i][j] += v[i][k] * b.v[k][j];
                if (MOD) {
                    if (ans.v[i][j] < 0)
                        ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
                    if (ans.v[i][j] >= MOD)
                        ans.v[i][j] %= MOD;
                }
            }
    }
    v = ans.v;
    return *this;
}

```

```

}
229 Matrix operator*=(T x) {
    int n = v.size(), m = v[0].size();
231 for (int i = 0; i < n; ++i)
    for (int j = 0; j < m; ++j) {
233 v[i][j] *= x;
        if (MOD) {
235 if (v[i][j] < 0)
            v[i][j] = (v[i][j] % MOD + MOD) % MOD;
237 if (v[i][j] >= MOD)
            v[i][j] %= MOD;
239 }
        }
241 return *this;
}
243 Matrix operator/(T x) {
    Matrix ans(*this);
245 int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
247 for (int j = 0; j < m; ++j) {
        if (MOD) {
249 ans.v[i][j] = modeq(x, (T)1, (T)MOD);
            if (ans.v[i][j] < 0)
251 ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
            if (ans.v[i][j] >= MOD)
253 ans.v[i][j] %= MOD;
        } else
255 ans.v[i][j] /= x;
    }
257 return ans;
}
259 Matrix operator/=(T x) {
    int n = v.size(), m = v[0].size();
261 for (int i = 0; i < n; ++i)
    for (int j = 0; j < m; ++j) {
263 if (MOD) {
        v[i][j] = modeq(x, (T)1, (T)MOD);
265 if (v[i][j] < 0)
            v[i][j] = (v[i][j] % MOD + MOD) % MOD;
267 if (v[i][j] >= MOD)
            v[i][j] %= MOD;
269 } else
        v[i][j] /= x;
    }
271 return *this;
}
273 Matrix operator%=(T p) {
    int n = v.size(), m = v[0].size();
275 for (int i = 0; i < n; ++i)
    for (int j = 0; j < m; ++j)
277 if (v[i][j] >= p)
        v[i][j] %= p;
279 return *this;
}
281 void gaussian() {
    int curi = 0;
    int n = v.size();
285 int m = v[0].size();
    for (int j = 0; j < m; j++) {
287 int i;
        for (i = curi; i < n; i++) {
289 if (MOD) {
            v[i][j] %= MOD;
291 }
            if (v[i][j]) {
293 break;
            }
295 }
        if (i >= n) {
297 continue;
        }
299 if (v[i][j] == 0)
            continue;
        for (int k = 0; k < m; k++) {
301 swap(v[i][k], v[curi][k]);
        }
303 for (int k = m - 1; k >= j; k--) {
        if (MOD) {
305 v[curi][k] = modeq(v[curi][j], (T)1, (T)MOD);
            v[curi][k] = (v[curi][k] % MOD + MOD) % MOD;
307 } else
            v[curi][k] /= v[curi][j];
        }
309 for (int i = 0; i < n; ++i) {
        if (i != curi) {
311 for (int k = m - 1; k >= j; k--) {
            v[i][k] -= v[curi][k] * v[i][j];
313 if (MOD) {
                v[i][k] = (v[i][k] % MOD + MOD) % MOD;
315 }
            }
317 }
    }
}

```

```

    }
    }
321 curi++;
    }
323 }
};

```

3. String

3.1. KMP

```

1 string s, t;
  int pmt[1000005];
3
  void init() {
5     for (int i = 1, j = 0; i < t.size(); i++) {
        while (j && t[j] ^ t[i]) {
7             j = pmt[j - 1];
        }
9         if (t[j] == t[i])
            j++;
11        pmt[i] = j;
    }
13 }
15 int kmp(string s) {
    int ret = 0;
17 for (int i = 0, j = 0; i < s.size(); i++) {
        while (j && s[i] ^ t[j]) {
19             j = pmt[j - 1];
        }
21        if (s[i] == t[j]) {
            j++;
23        }
        if (j == t.size()) {
25            ret++;
            j = pmt[j - 1];
27        }
    }
29    return ret;
}

```

3.2. Longest Palindrome

```

1
  #define int long long
  using namespace std;
3
5 string s;
  string t;
7 int n;
  int d[2000005];
9 int ans = 0;
11 signed main() {
    cin >> t;
13    n = t.size();
    for (int i = 0; i < 2 * n + 1; i++) {
15        if (i & 1 ^ 1) {
            s += '0';
17        } else {
            s += t[i / 2];
19        }
    }
21    n = s.size();
    d[0] = 1;
23    for (int i = 0, l = 0, r = 0; i < n; i++) {
        if (i > r) {
25            d[i] = 1;
            bool a = i + d[i] < n;
27            bool b = i - d[i] >= 0;
            bool c = (s[i + d[i]] == s[i - d[i]]);
29            while (a && b && c) {
                d[i]++;
31                a = i + d[i] < n;
                b = i - d[i] >= 0;
33                c = (s[i + d[i]] == s[i - d[i]]);
            }
35            l = i - d[i] + 1;
            r = i + d[i] - 1;
37        } else {
            int j = l + r - i;
39            if (j - d[j] + 1 > l) {
                d[i] = d[j];
41            } else {
                d[i] = r - i + 1;
43                a = i + d[i] < n;
                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 }

```

3.3. Z

```

1  #define int long long
2  using namespace std;
3
4  string s, t;
5  int ans = 0;
6
7  int z[2000005];
8
9  signed main() {
10     ios::sync_with_stdio(0);
11     cin.tie(0);
12     cout.tie(0);
13     cin >> s >> t;
14     s = t + '0' + s;
15     int n, m;
16     n = s.size();
17     m = t.size();
18     for (int i = 0, l = 0, r = 0; i < n; i++) {
19         if (z[i - l] < r - i + 1) {
20             z[i] = z[i - l];
21         } else {
22             z[i] = max(r - i + 1, (int)0);
23             while (i + z[i] < n && s[i + z[i]] == s[z[i]]) {
24                 z[i]++;
25             }
26             l = i;
27             r = i + z[i] - 1;
28             if (z[i] == m) {
29                 ans++;
30             }
31         }
32     }
33     cout << ans;
34 }

```

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[2000005];
6  int r[2000005];
7  int d[2000005];
8  int cycle[2000005];
9  int len[2000005];
10 int cnt = 0;
11 vector<int> v[2000005];
12 bitset<2000005> vis1;
13 bitset<2000005> vis2;
14
15 void findcycle(int x) {
16     while (!vis1[x]) {
17         vis1[x] = 1;
18         x = a[x];
19     }
20 }
21 cnt++;
22 cycle[x] = cnt;
23 r[x] = 0;
24 len[cnt] = 1;
25 int temp = a[x];

```

```

27 while (temp ^ x) {
28     r[temp] = len[cnt];
29     len[cnt]++;
30     cycle[temp] = cnt;
31     temp = a[temp];
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[1000005];
3  bitset<1000005> vis;
4  int dis[1000005];
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;

```



```

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;
        }
59         v[x].push_back(y);
        v[y].push_back(x);
61         c[x][y] = z;
    }
63     while (bfs()) {
        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--) {
        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         }
    }
    for (int i = 1; i <= m; i++) {
71         if (scc[i] ^ scc[i + m]) {
            op[scc[i]] = scc[i + m];
73             op[scc[i + m]] = scc[i];
        } else {
75             cout << "IMPOSSIBLE";
            exit(0);
77         }
    }
79     for (int i = 1; i <= 2 * m; i++) {
        for (int j : v[i]) {
81

```

```

83     if (scc[i] ^ scc[j]) {
        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();
    q.pop();
    if (!ans[k]) {
95         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;
    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;
    if (tree[index].a < ins.a)
21         swap(tree[index], ins);
    if (tree[index].y(mid) > ins.y(mid)) {
23         swap(tree[index], ins);
        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;
    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;
        int d2 = y.a - x.a;
        return d1 / d2;
    }

11 bool check(line x, line y, line z) {
    int I12 = intersect(x, y);
    int I23 = intersect(y, z);
    return I12 < I23;
13 }

15 void insert(int a, int b) {
    if (!dq.empty() && a == dq.back().a)
        return;
    while (dq.size() >= 2 &&
21         !check(dq[dq.size() - 2], dq[dq.size() - 1], {a, b})) {
        dq.pop_back();
    }
    dq.push_back({a, b});
23 }

25 void update(int x) {
    while (dq.size() >= 2 && dq[0].y(x) >= dq[1].y(x)) {
27         dq.pop_front();
    }
    dq.push_back({x, dq.back().y(x)});
31 }

33 int query(int x) {
    update(x);
    return dq.front().y(x);
35 }
};

```

6. Geometry

6.1. Intersect

```

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

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;
    if (k == 0)
        return 0;
    if (k > 0)
        return 1;
    if (k < 0)
        return -1;
15 }

17 bool intersect(point x, point y, point z, point w) {
    if (onseg(x, y, z) || onseg(x, y, w))
        return 1;
    if (onseg(z, w, x) || onseg(z, w, y))
        return 1;
    if (dir(y - x, z - x) * dir(y - x, w - x) == -1 &&
27     dir(z - w, x - w) * dir(z - w, y - w) == -1) {
        return 1;
    }
    return 0;
31 }

```

6.2. Inside

```

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

```


6.3. Minimum Euclidean Distance

```

1  #define int long long
3  #define pii pair<int, int>
4  using namespace std;
5
6  int n;
7  vector<pair<int, int>> v;
8  set<pair<int, int>> s;
9  int dd = LONG_LONG_MAX;
10
11 int dis(pii x, pii y) {
12     return (x.first - y.first) * (x.first - y.first) +
13            (x.second - y.second) * (x.second - y.second);
14 }
15
16 signed main() {
17     ios::sync_with_stdio(0);
18     cin.tie(0);
19     cout.tie(0);
20     cin >> n;
21     for (int i = 0; i < n; i++) {
22         int x, y;
23         cin >> x >> y;
24         x += 1000000000;
25         v.push_back({x, y});
26     }
27     sort(v.begin(), v.end());
28     int l = 0;
29     for (int i = 0; i < n; i++) {
30         int d = ceil(sqrt(dd));
31         while (l < i && v[l].first - v[i].first > d) {
32             s.erase({v[l].second, v[l].first});
33             l++;
34         }
35         auto x = s.lower_bound({v[i].second - d, 0});
36         auto y = s.upper_bound({v[i].second + d, 0});
37         for (auto it = x; it != y; it++) {
38             dd = min(dd, dis(*it->second, it->first}, v[i]));
39         }
40         s.insert({v[i].second, v[i].first});
41     }
42     cout << dd;
43 }

```

```

39     }
40     return max(query(L, mid, l, mid, index << 1),
41               query(mid + 1, R, mid + 1, r, index << 1 | 1));
42 }
43
44 void modify(int x, int val, int l = 1, int r = n, int index = 1) {
45     if (l == r) {
46         tree[index] = val;
47         return;
48     }
49     int mid = (l + r) >> 1;
50     if (x <= mid) {
51         modify(x, val, l, mid, index << 1);
52     } else {
53         modify(x, val, mid + 1, r, index << 1 | 1);
54     }
55     tree[index] = max(tree[index << 1], tree[index << 1 | 1]);
56 }
57
58 void dfs(int x, int pre) {
59     si[x] = 1;
60     for (int i : v[x]) {
61         if (i == pre)
62             continue;
63         p[i] = x;
64         d[i] = d[x] + 1;
65         dfs(i, x);
66         si[x] += si[i];
67     }
68 }
69
70 void dfs2(int x, int pre, int t) {
71     tp[x] = t;
72     st[x] = ++cnt;
73     int ma = 0;
74     for (int i : v[x]) {
75         if (i == pre)
76             continue;
77         if (si[i] > si[ma]) {
78             ma = i;
79         }
80     }
81     if (!ma)
82         return;
83     dfs2(ma, x, t);
84     for (int i : v[x]) {
85         if (i == pre || i == ma) {
86             continue;
87         }
88         dfs2(i, x, i);
89     }
90 }
91
92 int f(int x, int y) {
93     int ret = 0;
94     while (tp[x] ^ tp[y]) {
95         if (d[tp[x]] < d[tp[y]]) {
96             swap(x, y);
97         }
98         ret = max(ret, query(st[tp[x]], st[x]));
99         x = p[tp[x]];
100     }
101     if (d[x] > d[y])
102         swap(x, y);
103     ret = max(ret, query(st[x], st[y]));
104     return ret;
105 }
106
107 signed main() {
108     ios::sync_with_stdio(0);
109     cin.tie(0);
110     cout.tie(0);
111     cin >> n >> q;
112     for (int i = 1; i <= n; i++) {
113         cin >> a[i];
114     }
115     for (int i = 1; i < n; i++) {
116         int x, y;
117         cin >> x >> y;
118         v[x].push_back(y);
119         v[y].push_back(x);
120     }
121     dfs(1, 0);
122     dfs2(1, 0, 1);
123     for (int i = 1; i <= n; i++) {
124         b[st[i]] = a[i];
125     }
126     build();
127     while (q--) {
128         int mode, x, y;

```

7. Tree

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

```

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

```

```

129     cin >> mode >> x >> y;
131     if (mode == 1) {
133         modify(st[x], y);
135     } else {
137         cout << f(x, y) << " ";
139     }
141 }

```

7.2. LCA

```

1  #define int long long
3  using namespace std;
5  int n, q;
7  int a[200005][21];
9  int d[200005];
11 vector<int> v[200005];
13 void init() {
15     for (int j = 1; j < 21; j++) {
17         for (int i = 1; i <= n; i++) {
19             a[i][j] = a[a[i][j-1]][j-1];
21         }
23     }
25 }
27 void dfs(int x, int pre) {
29     for (int i : v[x]) {
31         if (i == pre) {
33             continue;
35         }
37         a[i][0] = x;
39         d[i] = d[x] + 1;
41         dfs(i, x);
43     }
45 }
47 int lca(int x, int y) {
49     while (d[x] ^ d[y]) {
51         if (d[x] < d[y]) {
53             swap(x, y);
55         }
57         int k = __lg(d[x] - d[y]);
59         x = a[x][k];
61     }
63     if (x == y) {
65         return x;
67     }
69     for (int i = 20; i >= 0; i--) {
71         if (a[x][i] != a[y][i]) {
73             x = a[x][i];
75             y = a[y][i];
77         }
79     }
81     return a[x][0];
83 }
85 signed main() {
87     ios::sync_with_stdio(0);
89     cin.tie(0);
91     cout.tie(0);
93     cin >> n >> q;
95     for (int i = 1; i < n; i++) {
97         int x, y;
99         cin >> x >> y;
101         v[x].push_back(y);
102         v[y].push_back(x);
103     }
104     dfs(1, 0);
105     init();
106     while (q--) {
107         int x, y;
108         cin >> x >> y;
109         int k = lca(x, y);
110         cout << (d[x] + d[y] - 2 * d[k]) << "\n";
111     }
112 }

```

```

7  double get(double x) {
9      double ret = 0;
11     double k = 1;
13     for (int i = 0; i <= n; i++) {
15         ret += k * a[i];
17         k *= x;
19     }
21     return -ret;
23 }
25 template <class T> T bi_search(T l, T r, T end) {
27     if (!check(r - end))
29         return r - end;
31     for (; r - l > end;) {
33         T mid = (l + r) / 2;
35         if (check(mid))
37             r = mid;
39         else
41             l = mid;
43     }
45     return l;
47 }
49 /*check gives 000000001111 find the last 0*/
51 template <class T> T tri_search(T l, T r, T end) {
53     T midl, midr;
55     for (;;) {
57         midl = (l + r) / 2;
59         midr = (midl + r) / 2;
61         if (midr - midl < end)
63             break;
65         if (get(midr) > get(midl))
67             r = midr;
69         else
71             l = midl;
73     }
75     for (; r - l > end;) {
77         midl = (l + r) / 2;
79         if (get(r) > get(l))
81             r = midl;
83         else
85             l = midl;
87     }
89     return l;
91 }
93 /*get gives the value, find the minimum*/
95 int main() {
97     cin >> n >> x >> y;
99     for (int i = n; i >= 0; i--) {
101         cin >> a[i];
102     }
103     cout << fixed << setprecision(7) << tri_search<double>(x, y, 1e-7) << "\n";
104 }

```

8. Misc

8.1. Tri Search

```

1  using namespace std;
3  int n;
5  double a[15], x, y;

```