

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

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 + treap[rson].sum + treap[index].val;
20 }
21 void push(int index) {
22     if (!treap[index].tag)
23         return;
24     swap(treap[index].lson, treap[index].rson);
25     int lson = treap[index].lson;
26     int rson = treap[index].rson;
27     treap[lson].tag ^= 1;
28     treap[rson].tag ^= 1;
29     treap[index].tag = 0;
30 }
31 pii split(int rk, int index) {

```

```

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

```

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;

```

```

25 tree[lson].tag += tree[index].tag;
26 tree[rson].tag += tree[index].tag;
27 tree[index].data = tree[index].rv();
28 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);

```

```

17 const double PI = acos(-1);
18 comp A[MAXN * 3], B[MAXN * 3], tmp[MAXN * 3], ans[MAXN * 3];
19 void fft(comp F[], int N, int sgn = 1) {
20     if (N == 1)
21         return;
22     memcpy(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     memcpy(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        x *= x;
15        x %= MOD;
16        y >>= 1;
17    }
18    return ret;
19 }
20
21 void ntt(int F[], int N, int sgn) {
22     int bit = __lg(N);
23     for (int i = 0; i < N; i++) {
24         rev[i] = (rev[i] >> 1) >> 1 | ((i & 1) << (bit - 1));
25         if (i < rev[i])
26             swap(F[i], F[rev[i]]);
27     }
28     for (int l = 1, t = 1; l < N; l <= 1, t++) {
29         int step = qpow(G, ((MOD - 1) >> t) * sgn + MOD - 1);
30         for (int i = 0; i < N; i += l << 1)
31             for (int k = i, cur = 1; k < i + l; k++) {
32                 int g = F[k], h = (ll)F[k + l] * cur % MOD;
33                 F[k] = (g + h) % MOD;
34                 F[k + l] = ((g - h) % MOD + MOD) % MOD;
35                 cur = (ll)cur * step % MOD;
36             }
37     }
38     if (sgn == -1) {
39         int invN = qpow(N, MOD - 2);
40         for (int i = 0; i < N; i++)
41             F[i] = (ll)F[i] * invN % MOD;
42     }
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
9 void gaussian(double a[105][105], int n, int m) {
    int curi = 0;
11    for (int j = 0; j < m; j++) {
        int i;
13        for (i = curi; i < n; i++) {
            if (a[i][j]) {
15                break;
            }
17        }
        if (a[i][j] == 0)
19            continue;
        for (int k = 0; k < m; k++) {
21            swap(a[i][k], a[curi][k]);
        }
23        for (int k = m - 1; k >= j; k--) {
            a[curi][k] /= a[curi][j];
25        }
        for (int i = 0; i < n; ++i) {
27            if (i != curi) {
                for (int k = m - 1; k >= j; k--) {
29                    a[i][k] -= a[curi][k] * a[i][j];
                }
31            }
33            curi++;
35        }
    }
}

```

2.4. Mu

```

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

```

2.5. Lucas

```

1 int fact[100005];
  int p;
3
  void init() {
5      fact[0] = 1;
      for (int i = 1; i <= p; i++) {
9          fact[i] = fact[i - 1] * i % p;
      }
11
12 int inv(int x, int p) {
    if (x == 1)
13         return 1;
    return (p - p / x) * inv(p % x, p) % p;
15 }
17
18 int c(int x, int y, int p) {
    if (x < y)
19         return 0;
    int k = fact[x] * inv(fact[y], p) % p;
21    return k * inv(fact[x - y], p) % p;
    }
23
24 int lucas(int x, int y, int p) {
    if (x == 0)
25         return 1;
    return lucas(x / p, y / p, p) % p * c(x % p, y % p, p) % p;
27 }
}

```

2.6. Inv

```

1 int exgcd(int a, int b, int &x, int &y) {
    if (b == 0) {

```

```

3         x = 1;
        y = 0;
5         return a;
    }
7     int d = exgcd(b, a % b, y, x);
    y -= x * (a / b);
9     return d;
}

11 int inv(int a, int p) {
13     int x, y;
    exgcd(a, p, x, y);
15     return (x % p + p) % p;
}

```

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 using namespace std;
3 template <class T> T extgcd(T a, T b, T &x, T &y) {
    if (!b) {
5         x = 1;
        y = 0;
        return a;
    }
7     T ans = extgcd(b, a % b, y, x);
    y -= a / b * x;
9     return ans;
}

11 template <class T> T modeq(T a, T b, T p) {
13     T x, y, d = extgcd(a, p, x, y);
    if (b % d)
15         return 0;
    return ((b / d * x) % p + p) % p;
17 }

18 template <class T> class Matrix {
19     static const T MOD = 1000000007;
21
22 public:
    vector<vector<T>> v;
23     Matrix(int n, int m, int identity) {
        v = vector<vector<T>>(n, vector<T>(m, 0));
25         if (identity)
            for (int i = 0, k = min(n, m); i < k; ++i)
27             v[i][i] = 1;
    }
    Matrix(Matrix &b) { v = b.v; }
    void in() {
33         n = v.size(), m = v[0].size();
        for (int i = 0; i < n; ++i)
35             for (int j = 0; j < m; ++j)
                scanf("%lld", &v[i][j]);
37     }
    Matrix(int n, int m) {
39         v = vector<vector<T>>(n, vector<T>(m, 0));
        in();
41     }
    void show() {
43         n = v.size(), m = v[0].size();
        for (int i = 0; i < n; ++i)
45             for (int j = 0; j < m; ++j)
                printf("%lld%c", v[i][j], " \n"[j == m - 1]);
47     }
    Matrix operator=(Matrix &b) {
49         v = b.v;
        return *this;
51     }
}

```

```

Matrix operator+(Matrix &b) {
    Matrix ans(*this);
    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);
    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);
    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);
    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) {
    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) {
    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) {
    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) {
    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);
    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;
}

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/(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) {
            if (MOD) {
                ans.v[i][j] *= modeq(x, 1LL, 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;
            } else
                ans.v[i][j] /= x;
        }
    return ans;
}
Matrix operator%=(T p) {
    n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j) {
            if (MOD) {
                v[i][j] *= modeq(x, 1LL, MOD);
                if (v[i][j] < 0)
                    v[i][j] = (v[i][j] % MOD + MOD) % MOD;
                if (v[i][j] >= MOD)
                    v[i][j] %= MOD;
            } else
                v[i][j] /= x;
        }
    return *this;
}
Matrix gaussian() {
    vector<vector<T>> a = v;
    int curi = 0;
    int n = v.size();
    int m = v[0].size();
    for (int j = 0; j < m; j++) {
        int i;
        for (i = curi; i < n; i++) {
            if (MOD) {
                a[i][j] %= MOD;
            }
            if (a[i][j]) {
                break;
            }
        }
        if (MOD) {
            a[i][j] %= MOD;
        }
        if (a[i][j] == 0)
            continue;
        for (int k = 0; k < m; k++) {
            swap(a[i][k], a[curi][k]);
        }
        for (int k = m - 1; k >= j; k--) {
            if (MOD) {
                a[curi][k] *= modeq(a[curi][j], 1LL, MOD);
                a[curi][k] = (a[curi][k] % MOD + MOD) % MOD;
            } else
                a[curi][k] /= a[curi][j];
        }
        for (int i = 0; i < n; ++i) {
            if (i != curi) {
                for (int k = m - 1; k >= j; k--) {
                    a[i][k] -= a[curi][k] * a[i][j];
                    if (MOD) {
                        a[i][k] = (a[i][k] % MOD + MOD) % MOD;
                    }
                }
            }
        }
        curi++;
    }
    return a;
}
};

```

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 ^ 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     }
56 }

```

```

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
3  using namespace std;

5  int n, q;
6  int a[200005];
7  int r[200005];
8  int d[200005];
9  int cycle[200005];
10 int len[200005];
11 int cnt = 0;
12 vector<int> v[200005];
13 bitset<200005> vis1;
14 bitset<200005> vis2;

15 void findcycle(int x) {
17     while (!vis1[x]) {
18         vis1[x] = 1;
19         x = a[x];
20     }
21     cnt++;
22     cycle[x] = cnt;
23     r[x] = 0;
24     len[cnt] = 1;
25     int temp = a[x];
26     while (temp ^ x) {
27         r[temp] = len[cnt];
28         len[cnt]++;
29         cycle[temp] = cnt;
30         temp = a[temp];
31     }
32 }

33 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 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 signed main() {
51     ios::sync_with_stdio(0);
52     cin.tie(0);
53     cout.tie(0);
54     cin >> n;
55     for (int i = 1; i <= n; i++) {
56         cin >> a[i];
57         v[i].push_back(a[i]);
58         v[a[i]].push_back(i);
59     }
60     for (int i = 1; i <= n; i++) {
61         if (!vis2[i]) {
62             findcycle(i);
63             dfs(i);
64         }
65     }
66     for (int i = 1; i <= n; i++) {
67         if (!cycle[i] && !r[i]) {
68             dfs2(i);
69         }
70     }
71     for (int i = 1; i <= n; i++) {

```

```

75     cout << d[i] + len[cycle[i]] << " ";
76 }

```

4.2. Dijkstra

```

1  int n, m;
2  vector<pair<int, int>> v[100005];
3  bitset<100005> vis;
4  int dis[100005];

5  void dijkstra(int x) {
6      priority_queue<pair<int, int>, vector<pair<int, int>>,
7          greater<pair<int, int>>>
8          pq;
9      memset(dis, 0x3f, sizeof(dis));
10     dis[x] = 0;
11     pq.push({0, x});
12     while (!pq.empty()) {
13         pair<int, int> now = pq.top();
14         pq.pop();
15         if (vis[now.second])
16             continue;
17         vis[now.second] = 1;
18         for (auto [i, w] : v[now.second]) {
19             if (vis[i])
20                 continue;
21             if (dis[now.second] + w < dis[i]) {
22                 dis[i] = dis[now.second] + w;
23                 pq.push({dis[i], i});
24             }
25         }
26     }
27 }

```

4.3. MaximumFlow

```

1  #define int long long
3  using namespace std;

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

11 bool bfs() {
12     memset(head, 0, sizeof(head));
13     memset(lv, 0, sizeof(lv));
14     queue<int> q;
15     q.push(1);
16     while (!q.empty()) {
17         int now = q.front();
18         q.pop();
19         if (now == n)
20             continue;
21         for (int i : v[now]) {
22             if (i != 1 && c[now][i] && !lv[i]) {
23                 lv[i] = lv[now] + 1;
24                 q.push(i);
25             }
26         }
27     }
28     return lv[n];
29 }

30 int dfs(int x, int flow) {
31     int ret = 0;
32     if (x == n)
33         return flow;
34     for (int i = head[x]; i < v[x].size(); i++) {
35         int y = v[x][i];
36         head[x] = y;
37         if (c[x][y] && lv[y] == lv[x] + 1) {
38             int d = dfs(y, min(flow, c[x][y]));
39             flow -= d;
40             c[x][y] -= d;
41             c[y][x] += d;
42             ret += d;
43         }
44     }
45     return ret;
46 }

47 signed main() {
48     cin >> n >> m;
49     while (m--) {
50         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;
  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])
            continue;
        if (d[i]) {
15             low[x] = min(low[x], d[i]);
        } else {
17             dfs(i);
            low[x] = min(low[x], low[i]);
19         }
    }
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
  using namespace std;
3
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)
        return x - m;
    return x + m;
21 }
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;
        if (d[i]) {
31             low[x] = min(low[x], d[i]);
        } else {
33             dfs(i);
            low[x] = min(low[x], low[i]);
35         }
    }
37 }

```

```

    if (d[x] == low[x]) {
        now++;
        while (!s.empty()) {
41             int k = s.top();
            s.pop();
43             scc[k] = now;
            if (k == x)
                break;
45         }
    }
47 }
49
signed main() {
51     ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
53     cin >> n >> m;
    while (n--) {
55         char a, b;
        int x, y;
57         cin >> a >> x >> b >> y;
        if (a == '-')
            x = no(x);
59         if (b == '-')
            y = no(y);
61         v[no(x)].push_back(y);
        v[no(y)].push_back(x);
63     }
    for (int i = 1; i <= 2 * m; i++) {
65         if (!d[i]) {
            dfs(i);
67         }
    }
    for (int i = 1; i <= m; i++) {
71         if (scc[i] ^ scc[i + m]) {
            op[scc[i]] = scc[i + m];
            op[scc[i + m]] = scc[i];
73         } else {
            cout << "IMPOSSIBLE";
            exit(0);
75         }
    }
    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             }
        }
    }
    for (int i = 1; i <= now; i++) {
87         if (!ind[i]) {
            q.push(i);
89         }
    }
    while (!q.empty()) {
91         int k = q.front();
        q.pop();
93         if (ans[k]) {
            ans[op[k]] = 2;
95         }
        for (int i : v2[k]) {
101             ind[i]--;
            if (!ind[i]) {
                q.push(i);
103             }
        }
    }
    for (int i = 1; i <= m; i++) {
107         if (ans[scc[i]] == 1) {
            cout << "+ ";
109         } else {
            cout << "- ";
111         }
    }
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];
int f[200005];
int dp[200005];

void update(line ins, int l = 1, int r = 1e6, int index = 1) {
    if (l == r) {
        if (ins.y(l) < tree[index].y(l)) {
            tree[index] = ins;
        }
        return;
    }
    int mid = (l + r) >> 1;
    if (tree[index].a < ins.a)
        swap(tree[index], ins);
    if (tree[index].y(mid) > ins.y(mid)) {
        swap(tree[index], ins);
        update(ins, l, mid, index << 1);
    } else {
        update(ins, mid + 1, r, index << 1 | 1);
    }
}

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

```

5.2. CHO

```

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

struct CHO {
    deque<line> dq;
    int intersect(line x, line y) {
        int d1 = x.b - y.b;
        int d2 = y.a - x.a;
        return d1 / d2;
    }
    bool check(line x, line y, line z) {
        int I12 = intersect(x, y);
        int I23 = intersect(y, z);
        return I12 < I23;
    }
    void insert(int a, int b) {
        if (!dq.empty() && a == dq.back().a)
            return;
        while (dq.size() >= 2 &&
            !check(dq[dq.size() - 2], dq[dq.size() - 1], {a, b})) {
            dq.pop_back();
        }
        dq.push_back({a, b});
    }
    void update(int x) {
        while (dq.size() >= 2 && dq[0].y(x) >= dq[1].y(x)) {
            dq.pop_front();
        }
    }
    int query(int x) {
        update(x);
        return dq.front().y(x);
    }
};

```

6. Geometry

6.1. Intersect

```

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

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

```

```

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

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

```

6.2. Inside

```

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;
}

```

6.3. Minimum Euclidean Distance

```

#define int long long
#define pii pair<int, int>
using namespace std;

int n;
vector<pair<int, int>> v;
set<pair<int, int>> s;
int dd = LONG_LONG_MAX;

int dis(pii x, pii y) {
    return (x.first - y.first) * (x.first - y.first) +
        (x.second - y.second) * (x.second - y.second);
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin >> n;
    for (int i = 0; i < n; i++) {
        int x, y;
        cin >> x >> y;
        x += 1000000000;
        v.push_back({x, y});
    }
    sort(v.begin(), v.end());
    int l = 0;
    for (int i = 0; i < n; i++) {
        int d = ceil(sqrt(dd));
        while (l < i && v[i].first - v[l].first > d) {
            s.erase({v[l].second, v[l].first});
            l++;
        }
        auto x = s.lower_bound({v[i].second - d, 0});
        auto y = s.upper_bound({v[i].second + d, 0});
        for (auto it = x; it != y; it++) {
            dd = min(dd, dis(*it, v[i]));
        }
        s.insert({v[i].second, v[i].first});
    }
    cout << dd;
}

```


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;
9  int a[200005];
11 int st[200005];
13 int tp[200005];
15 int p[200005];
17 int cnt = 0;
19 int d[200005];
21 int si[200005];
23 vector<int> v[200005];
25 int b[200005];
27
29 void build(int l = 1, int r = n, int index = 1) {
31     if (l == r) {
33         tree[index] = b[l];
35         return;
37     }
39     int mid = (l + r) >> 1;
41     build(l, mid, index << 1);
43     build(mid + 1, r, index << 1 | 1);
45     tree[index] = max(tree[index << 1], tree[index << 1 | 1]);
47 }
49
51 int query(int L, int R, int l = 1, int r = n, int index = 1) {
53     if (L == l && R == r) {
55         return tree[index];
57     }
59     int mid = (l + r) >> 1;
61     if (R <= mid) {
63         return query(L, R, l, mid, index << 1);
65     }
67     if (L > mid) {
69         return query(L, R, mid + 1, r, index << 1 | 1);
71     }
73     return max(query(L, mid, l, mid, index << 1),
75               query(mid + 1, R, mid + 1, r, index << 1 | 1));
77 }
79
81 void modify(int x, int val, int l = 1, int r = n, int index = 1) {
83     if (l == r) {
85         tree[index] = val;
87         return;
89     }
91     int mid = (l + r) >> 1;
93     if (x <= mid) {
95         modify(x, val, l, mid, index << 1);
97     }
99     else {
101         modify(x, val, mid + 1, r, index << 1 | 1);
103     }
105     tree[index] = max(tree[index << 1], tree[index << 1 | 1]);
107 }
109
111 void dfs(int x, int pre) {
113     si[x] = 1;
115     for (int i : v[x]) {
117         if (i == pre)
119             continue;
121         p[i] = x;
123         d[i] = d[x] + 1;
125         dfs(i, x);
127         si[x] += si[i];
129     }
131 }
133
135 void dfs2(int x, int pre, int t) {
137     tp[x] = t;
139     st[x] = ++cnt;
141     int ma = 0;
143     for (int i : v[x]) {
145         if (i == pre)
147             continue;
149         if (si[i] > si[ma]) {
151             ma = i;
153         }
155     }
157     if (!ma)
159         return;
161     dfs2(ma, x, t);
163     for (int i : v[x]) {
165         if (i == pre || i == ma)
167             continue;

```

```

87         continue;
89     }
91     dfs2(i, x, i);
93 }
95
97 int f(int x, int y) {
99     int ret = 0;
101     while (tp[x] ^ tp[y]) {
103         if (d[tp[x]] < d[tp[y]]) {
105             swap(x, y);
107         }
109         ret = max(ret, query(st[tp[x]], st[x]));
111         x = p[tp[x]];
113     }
115     if (d[x] > d[y])
117         swap(x, y);
119     ret = max(ret, query(st[x], st[y]));
121     return ret;
123 }
125
127 signed main() {
129     ios::sync_with_stdio(0);
131     cin.tie(0);
133     cout.tie(0);
135     cin >> n >> q;
137     for (int i = 1; i <= n; i++) {
139         cin >> a[i];
141     }
143     for (int i = 1; i < n; i++) {
145         int x, y;
147         cin >> x >> y;
149         v[x].push_back(y);
151         v[y].push_back(x);
153     }
155     dfs(1, 0);
157     dfs2(1, 0, 1);
159     for (int i = 1; i <= n; i++) {
161         b[st[i]] = a[i];
163     }
165     build();
167     while (q--) {
169         int mode, x, y;
171         cin >> mode >> x >> y;
173         if (mode == 1) {
175             modify(st[x], y);
177         } else {
179             cout << f(x, y) << " ";
181         }
183     }
185 }

```

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
15 void init() {
17     for (int j = 1; j < 21; j++) {
19         for (int i = 1; i <= n; i++) {
21             a[i][j] = a[a[i][j - 1]][j - 1];
23         }
25     }
27 }
29
31 void dfs(int x, int pre) {
33     for (int i : v[x]) {
35         if (i == pre)
37             continue;
39         a[i][0] = x;
41         d[i] = d[x] + 1;
43         dfs(i, x);
45     }
47 }
49
51 int lca(int x, int y) {
53     while (d[x] ^ d[y]) {
55         if (d[x] < d[y]) {
57             swap(x, y);
59         }
61         int k = __lg(d[x] - d[y]);
63         x = a[x][k];
65     }
67 }

```

```

37  if (x == y) {
38      return x;
39  }
40  for (int i = 20; i >= 0; i--) {
41      if (a[x][i] != a[y][i]) {
42          x = a[x][i];
43          y = a[y][i];
44      }
45  }
46  return a[x][0];
47 }

49 signed main() {
50     ios::sync_with_stdio(0);
51     cin.tie(0);
52     cout.tie(0);
53     cin >> n >> q;
54     for (int i = 1; i < n; i++) {
55         int x, y;
56         cin >> x >> y;
57         v[x].push_back(y);
58         v[y].push_back(x);
59     }
60     dfs(1, 0);
61     init();
62     while (q--) {
63         int x, y;
64         cin >> x >> y;
65         int k = lca(x, y);
66         cout << (d[x] + d[y] - 2 * d[k]) << "\n";
67     }
68 }

```

```

55     cin >> n >> x >> y;
56     for (int i = n; i >= 0; i--) {
57         cin >> a[i];
58     }
59     cout << fixed << setprecision(7) << tri_search<double>(x, y, 1e-7) << "\n";

```

8. Misc

8.1. Tri Search

```

1  using namespace std;
2  int n;
3  double a[15], x, y;
4
5  double get(double x) {
6      double ret = 0;
7      double k = 1;
8      for (int i = 0; i <= n; i++) {
9          ret += k * a[i];
10         k *= x;
11     }
12     return -ret;
13 }
14
15 template <class T> T bi_search(T l, T r, T end) {
16     if (!check(r - end))
17         return r - end;
18     for (; r - l > end; ) {
19         T mid = (l + r) / 2;
20         if (check(mid))
21             r = mid;
22         else
23             l = mid;
24     }
25     return l;
26 }
27 /*check gives 000000001111 find the last 0*/
28
29 template <class T> T tri_search(T l, T r, T end) {
30     T midl, midr;
31     for (;;) {
32         midl = (l + r) / 2;
33         midr = (midl + r) / 2;
34         if (midr - midl < end)
35             break;
36         if (get(midr) > get(midl))
37             r = midr;
38         else
39             l = midl;
40     }
41     for (; r - l > end; ) {
42         midl = (l + r) / 2;
43         if (get(r) > get(l))
44             r = midl;
45         else
46             l = midl;
47     }
48     return l;
49 }
50 /*get gives the value, find the minimum*/
51
52 int main() {

```