

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 CRT
- 2.8 Generator
- 2.9 Count Primes
- 2.10 Pollard Rho
- 2.11 Formula
 - 2.11.1 Dirichlet Convolution
 - 2.11.2 Burnside's Lemma
 - 2.11.3 Pick Theorem
 - 2.11.4 Fermat's Little Theorem
 - 2.11.5 Wilson's Theorem
 - 2.11.6 Legendre Theorem
 - 2.11.7 Kummer Theorem
 - 2.11.8 ext-Kummer Theorem
 - 2.11.9 Factorial with mod
 - 2.11.10 Properties of nCr with mod
 - 2.11.11 ext-Lucas' Theorem
 - 2.11.12 Catalan Number
 - 2.11.13 modinv table
- 2.12 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
- 6.4 Convex Hull

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 }
33
34 pii split(int rk, int index) {
35     if (!index)
36         return {0, 0};
37     push(index);
38     int lson = treap[index].lson;
39     int rson = treap[index].rson;
40     if (rk <= treap[lson].si) {
41         pii temp = split(rk, lson);
42         treap[index].lson = temp.second;
43         update(index);
44         return {temp.first, index};
45     } else {
46         pii temp = split(rk - treap[lson].si - 1, rson);
47         treap[index].rson = temp.first;
48         update(index);
49         return {index, temp.second};
50     }
51 }
52
53 int merge(int x, int y) {
54     if (!x && !y)
55         return 0;
56     if (!x && y)
57         return y;
58     if (x && !y)
59         return x;
60     push(x);
61     push(y);
62     if (treap[x].prio < treap[y].prio) {
63         treap[x].rson = merge(treap[x].rson, y);
64         update(x);
65         return x;
66     } else {
67         treap[y].lson = merge(x, treap[y].lson);
68         update(y);
69         return y;
70     }
71 }
72
73 void insert(int x, int v) {
74     pii temp = split(x - 1, root);
75     cnt++;
76     treap[cnt].val = v;
77     update(cnt);
78     temp.first = merge(temp.first, cnt);
79     root = merge(temp.first, temp.second);
80 }
81
82 int query(int l, int r) {
83     pii R = split(r, root);
84     pii L = split(l - 1, R.first);
85     int ret = treap[L.second].sum;
86     R.first = merge(L.first, L.second);
87     root = merge(R.first, R.second);
88     return ret;
89 }
90
91 void modify(int l, int r) {
92     pii R = split(r, root);
93     pii L = split(l - 1, R.first);
94     treap[L.second].tag ^= 1;
95     R.first = merge(L.first, L.second);
96     root = merge(R.first, R.second);
97 }
```

1.2. Dynamic Segment Tree

```
1 #define int long long
2 using namespace std;
3
4 int n, q;
5 struct node {
```

```

7   int data, lson, rson, tag;
   int rv() { return data + tag; }
9   };

11  node tree[20000005];
   int a[200005];
13  int now = 1;
   int mx = 1000000005;

15  void push(int index) {
17      if (!tree[index].lson) {
19          tree[index].lson = ++now;
21          if (!tree[index].rson) {
23              tree[index].rson = ++now;
25              int lson = tree[index].lson;
27              int rson = tree[index].rson;
29              tree[lson].tag += tree[index].tag;
31              tree[rson].tag += tree[index].tag;
33              tree[index].data = tree[index].rv();
35              tree[index].tag = 0;
37          }
39      }
41      if (!tree[index].lson) {
43          tree[index].lson = ++now;
45          if (!tree[index].rson) {
47              tree[index].rson = ++now;
49              int lson = tree[index].lson;
51              int rson = tree[index].rson;
53              tree[lson].tag += tree[index].tag;
55              tree[rson].tag += tree[index].tag;
57              tree[index].data = tree[index].rv();
59              tree[index].tag = 0;
61          }
63      }
65      if (R <= mid) {
67          modify(l, mid, L, R, val, lson);
69      } else if (L > mid) {
71          modify(mid + 1, r, L, R, val, rson);
73      } else {
75          modify(l, mid, L, mid, val, lson);
77          modify(mid + 1, r, mid + 1, R, val, rson);
79      }
81      tree[index].data = tree[lson].rv() + tree[rson].rv();
83  }

85  int query(int l, int r, int L, int R, int index) {
87      // cout << L << " " << R << "\n";
89      if (l == L && r == R) {
91          return tree[index].rv();
93      }
95      int mid = (l + r) >> 1;
97      push(index);
99      int lson = tree[index].lson;
101     int rson = tree[index].rson;
103     if (R <= mid) {
105         return query(l, mid, L, R, lson);
107     }
109     if (L > mid) {
111         return query(mid + 1, r, L, R, rson);
113     }
115     return query(l, mid, L, mid, lson) + query(mid + 1, r, mid + 1, R, rson);
117 }

119 signed main() {
121     ios::sync_with_stdio(0);
123     cin.tie(0);
125     cout.tie(0);
127     cin >> n >> q;
129     for (int i = 1; i <= n; i++) {
131         cin >> a[i];
133         modify(1, mx, a[i], a[i], 1, 1);
135     }
137     while (q--) {
139         char mode;
141         int x, y;
143         cin >> mode;
145         if (mode == '?') {
147             cin >> x >> y;
149             cout << query(1, mx, x, y, 1) << "\n";
151         } else {
153             cin >> x >> y;
155             modify(1, mx, a[x], a[x], -1, 1);
157             a[x] = y;
159             modify(1, mx, a[x], a[x], 1, 1);
161         }
163     }
165 }

```

2. Math

2.1. FFT

```

1   using namespace std;
3   inline int read() {
5       int ans = 0;
7       char c = getchar();
9       while (!isdigit(c))
10          c = getchar();
11       while (isdigit(c)) {
13          ans = ans * 10 + c - '0';
15          c = getchar();
17       }
19       return ans;
21   }
23   typedef complex<double> comp;
25   const int MAXN = 1000005;
27   const comp I(0, 1);
29   const double PI = acos(-1);
31   comp A[MAXN * 3], B[MAXN * 3], tmp[MAXN * 3], ans[MAXN * 3];
33   void fft(comp F[], int N, int sgn = 1) {
35       if (N == 1)
37           return;
39       memcopy(tmp, F, sizeof(comp) * N);
41       for (int i = 0; i < N; i++)
43           *(i % 2 ? F + i / 2 + N / 2 : F + i / 2) = tmp[i];
45       fft(F, N / 2, sgn), fft(F + N / 2, N / 2, sgn);
47       comp *G = F, *H = F + N / 2;
49       comp cur = 1, step = exp(2 * PI / N * sgn * I);
51       for (int k = 0; k < N / 2; k++) {
53         tmp[k] = G[k] + cur * H[k];
55         tmp[k + N / 2] = G[k] - cur * H[k];
57         cur *= step;
59     }
61     memcopy(F, tmp, sizeof(comp) * N);
63 }
65 int main() {
67     int n = read(), m = read(), N = 1 << __lg(n + m + 1) + 1;
69     for (int i = 0; i <= n; i++)
71         A[i] = read();
73     for (int i = 0; i <= m; i++)
75         B[i] = read();
77     fft(A, N), fft(B, N);
79     for (int i = 0; i < N; i++)
81         ans[i] = A[i] * B[i];
83     fft(ans, N, -1);
85     for (int i = 0; i <= n + m; i++)
87         printf("%d ", int(ans[i].real() / N + 0.1));
89     return 0;
91 }

```

2.2. NTT

```

1   #define ll long long
3   using namespace std;
5   const int MAXN = 1000005;
7   const int MOD = 998244353, G = 3;
9   int rev[MAXN * 3];

11  int qpow(int x, int y) {
13     int ret = 1;
15     while (y) {
17         if (y & 1)
19             ret *= x;
21         x *= x;
23         y >>= 1;
25     }
27     return ret;
29 }

31  void ntt(int F[], int N, int sgn) {
33     int bit = __lg(N);
35     for (int i = 0; i < N; i++) {
37         rev[i] = (rev[i] >> 1) | ((i & 1) << (bit - 1));
39         if (i < rev[i])
41             swap(F[i], F[rev[i]]);
43     }
45     for (int l = 1, t = 1; l < N; l <= t, t *= 2) {
47         int step = qpow(G, ((MOD - 1) >> t) * sgn + MOD - 1);
49         for (int i = 0; i < N; i += t) {
51             for (int k = i, cur = 1; k < i + t; k++) {
53                 int g = F[k], h = (ll)F[k + l] * cur % MOD;
55                 F[k] = (g + h) % MOD;
57                 F[k + l] = (g - h) % MOD;
59                 cur = cur * step % MOD;
61             }
63         }
65     }
67 }

```

```

    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;
}
}

```

2.3. Gaussian-Jordan

```

1  #define int long long
3  using namespace std;

5  int n;
6  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         }
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     }
36 }

```

2.4. Mu

```

1  vector<int> prime;
2  bitset<1000005> vis;
3  int n;
4  int mu[1000005];

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

```

2.5. Lucas

```

1  int fact[100005];
2  int p;

3  void init() {
4      fact[0] = 1;
5      for (int i = 1; i <= p; i++) {
6          fact[i] = fact[i - 1] * i % p;
7      }
8  }

9  int inv(int x, int p) {
10     if (x == 1)

```

```

13     return 1;
14     return (p - p / x) * inv(p % x, p) % p;
15 }

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 int lucas(int x, int y, int p) {
24     if (x == 0)
25         return 1;
26     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) {
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 int inv(int a, int p) {
12     int x, y;
13     exgcd(a, p, x, y);
14     return (x % p + p) % p;
15 }

```

2.7. CRT

```

1  #define int long long
3  using namespace std;

5  int n;
6  int a[15];
7  int b[15];
8  int mul = 1;

9  void exgcd(int a, int b, int &x, int &y) {
10     if (b == 0) {
11         x = 1;
12         y = 0;
13         return;
14     }
15     exgcd(b, a % b, y, x);
16     y -= (a / b) * x;
17 }

18 int inv(int a, int p) {
19     int x, y;
20     exgcd(a, p, x, y);
21     return x;
22 }

23 int ans = 0;

24 signed main() {
25     cin >> n;
26     for (int i = 1; i <= n; i++) {
27         cin >> a[i] >> b[i];
28         mul *= a[i];
29     }
30     for (int i = 1; i <= n; i++) {
31         ans += inv(mul / a[i], a[i]) * (mul / a[i]) % mul * b[i] % mul;
32         ans %= mul;
33     }
34     ans = (ans + mul) % mul;
35     cout << ans;
36 }

```

2.8. Generator

```

1  #define int long long
3  using namespace std;

5  int t;
6  int n, d;
7  bitset<1000005> exist;
8  bitset<1000005> vis;
9  vector<int> prime;

```

```

11 int phi[1000005];
12
13 void init() {
14     phi[1] = 1;
15     for (int i = 2; i <= 1000000; i++) {
16         if (!vis[i]) {
17             prime.push_back(i);
18             phi[i] = i - 1;
19         }
20         for (int j : prime) {
21             if (i * j > 1000000)
22                 break;
23             vis[i * j] = 1;
24             if (i % j == 0) {
25                 phi[i * j] = phi[i] * j;
26                 break;
27             } else {
28                 phi[i * j] = phi[i] * phi[j];
29             }
30         }
31     }
32     exist[2] = exist[4] = 1;
33     for (int i : prime) {
34         if (i == 2)
35             continue;
36         for (int j = i; j <= 1000000; j *= i) {
37             exist[j] = 1;
38             if (j * 2 <= 1000000) {
39                 exist[j * 2] = 1;
40             }
41         }
42     }
43
44     vector<int> factors(int x) {
45         vector<int> v;
46         for (int i = 1; i * i <= x; i++) {
47             if (x % i == 0) {
48                 v.push_back(i);
49                 if (i * i != x) {
50                     v.push_back(x / i);
51                 }
52             }
53         }
54         return v;
55     }
56
57     int f(int x, int y, int mod) {
58         int ret = 1;
59         while (y) {
60             if (y & 1) {
61                 ret *= x;
62                 ret %= mod;
63             }
64             x *= x;
65             x %= mod;
66             y >>= 1;
67         }
68         return (ret % mod + mod) % mod;
69     }
70
71     vector<int> findroot(int x) {
72         vector<int> ret;
73         if (!exist[x])
74             return ret;
75         int phix = phi[x];
76         vector<int> fact = factors(phix);
77         int fst;
78         for (int i = 1; i++) {
79             if (__gcd(i, x) != 1)
80                 continue;
81             bool ok = 1;
82             for (int j : fact) {
83                 if (j != phix && f(i, j, x) == 1) {
84                     ok = 0;
85                     break;
86                 }
87             }
88             if (ok) {
89                 fst = i;
90                 break;
91             }
92         }
93         int now = fst;
94         // cout << fst << "\n";
95         for (int i = 1; i <= phix; i++) {
96             if (__gcd(i, phix) == 1) {
97                 ret.push_back(now);
98             }
99             now *= fst;

```

```

101     }
102     now %= x;
103     return ret;
104 }
105
106 signed main() {
107     ios::sync_with_stdio(0);
108     cin.tie(0);
109     cout.tie(0);
110     init();
111     cin >> t;
112     while (t--) {
113         cin >> n >> d;
114         vector<int> v = findroot(n);
115         sort(v.begin(), v.end());
116         cout << v.size() << "\n";
117         for (int i = 0; i < v.size(); i++) {
118             if (i % d == d - 1) {
119                 cout << v[i] << " ";
120             }
121         }
122         cout << "\n";
123     }

```

2.9. Count Primes

```

1 using namespace std;
2
3 using i64 = long long;
4 i64 count_pi(i64 N) {
5     if (N <= 1)
6         return 0;
7     int v = sqrt(N + 0.5);
8     int n_4 = sqrt(v + 0.5);
9     int T = min((int)sqrt(n_4) * 2, n_4);
10    int K = pow(N, 0.625) / log(N) * 2;
11    K = max(K, v);
12    K = min<i64>(K, N);
13    int B = N / K;
14    B = N / (N / B);
15    B = min<i64>(N / (N / B), K);
16
17    vector<i64> l(v + 1);
18    vector<int> s(K + 1);
19    vector<bool> e(K + 1);
20    vector<int> w(K + 1);
21    for (int i = 1; i <= v; ++i)
22        l[i] = N / i - 1;
23    for (int i = 1; i <= v; ++i)
24        s[i] = i - 1;
25
26    const auto div = [](i64 n, int d) -> int { return double(n) / d; };
27    int p;
28    for (p = 2; p <= T; ++p)
29        if (s[p] != s[p - 1]) {
30            i64 M = N / p;
31            int t = v / p, t0 = s[p - 1];
32            for (int i = 1; i <= t; ++i)
33                l[i] -= l[i * p] - t0;
34            for (int i = t + 1; i <= v; ++i)
35                l[i] -= s[div(M, i)] - t0;
36            for (int i = v, j = t; j >= p; --j)
37                for (int l = j * p; i >= l; --i)
38                    s[i] -= s[j] - t0;
39            for (int i = p * p; i <= K; i += p)
40                e[i] = 1;
41        }
42    e[1] = 1;
43    int cnt = 1;
44    vector<int> roughs(B + 1);
45    for (int i = 1; i <= B; ++i)
46        if (!e[i])
47            roughs[cnt++] = i;
48    roughs[cnt] = 0x7fffffff;
49    for (int i = 1; i <= K; ++i)
50        w[i] = e[i] + w[i - 1];
51    for (int i = 1; i <= K; ++i)
52        s[i] = w[i] - w[i - (i & -i)];
53
54    const auto query = [&](int x) -> int {
55        int sum = x;
56        while (x)
57            sum -= s[x], x ^= x & -x;
58        return sum;
59    };
60
61    const auto add = [&](int x) -> void {
62        e[x] = 1;
63        while (x <= K)
64            ++s[x], x += x & -x;

```

```

};
cnt = 1;
for (; p <= n_4; ++p)
    if (!e[p]) {
        i64 q = i64(p) * p, M = N / p;
        while (cnt < q)
            w[cnt] = query(cnt), cnt++;
        int t1 = B / p, t2 = min<i64>(B, M / q), t0 = query(p - 1);
        int id = 1, i = 1;
        for (; i <= t1; i = roughs[++id])
            l[i] -= l[i * p] - t0;
        for (; i <= t2; i = roughs[++id])
            l[i] -= query(div(M, i)) - t0;
        for (; i <= B; i = roughs[++id])
            l[i] -= w[div(M, i)] - t0;
        for (int i = q; i <= K; i += p)
            if (!e[i])
                add(i);
    }
while (cnt <= v)
    w[cnt] = query(cnt), cnt++;

vector<int> primes;
primes.push_back(1);
for (int i = 2; i <= v; ++i)
    if (!e[i])
        primes.push_back(i);
l[1] += i64(w[v] + w[n_4] - 1) * (w[v] - w[n_4]) / 2;
for (int i = w[n_4] + 1; i <= w[B]; ++i)
    l[i] -= l[primes[i]];
for (int i = w[B] + 1; i <= w[v]; ++i)
    l[i] -= query(N / primes[i]);
for (int i = w[n_4] + 1; i <= w[v]; ++i) {
    int q = primes[i];
    i64 M = N / q;
    int e = w[M / q];
    if (e <= i)
        break;
    l[1] += e - i;
    i64 t = 0;
    int m = w[sqrt(M + 0.5)];
    for (int k = i + 1; k <= m; ++k)
        t += w[div(M, primes[k])];
    l[1] += 2 * t - (i + m) * (m - i);
}
return l[1];
}

```

2.10. Pollard Rho

```

1 using namespace std;
2 #define LL long long
3 #define uLL __uint128_t
4 #define sub(a, b) ((a) < (b) ? (b) - (a) : (a) - (b))
5 template <class T, class POW> void fastpow(T x, POW n, POW p, T &ans) {
6     for (; n >= 1) {
7         if (n & 1) {
8             ans *= x;
9             ans %= p;
10        }
11        x *= x;
12        x %= p;
13    }
14 }
15 /*input x, n, p, ans, will modify ans to x ^ n % p
16 the first is x, ans and the second is n, p (LL or __int128)
17 */
18 uLL pri[7] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
19 // int p[3]={2,7,61};/*2^32*/
20 bool check(const uLL x, const uLL p) {
21     uLL d = x - 1, ans = 1;
22     fastpow(p, d, x, ans);
23     if (ans != 1)
24         return 1;
25     for (; !(d & 1);) {
26         d >= 1;
27         ans = 1;
28         fastpow(p, d, x, ans);
29         if (ans == x - 1)
30             return 0;
31         else if (ans != 1)
32             return 1;
33     }
34     return 0;
35 }
36 bool miller_rabin(const uLL x) {
37     if (x == 1)
38         return 0;
39     for (auto e : pri) {

```

```

41         if (e >= x)
42             return 1;
43         if (check(x, e))
44             return 0;
45     }
46     return 1;
47 }
48 template <class T> T gcd(T a, T b) {
49     if (!a)
50         return b;
51     if (!b)
52         return a;
53     if (a & b & 1)
54         return gcd(sub(a, b), min(a, b));
55     if (a & 1)
56         return gcd(a, b >> 1);
57     if (b & 1)
58         return gcd(a >> 1, b);
59     return gcd(a >> 1, b >> 1) << 1;
60 }
61 /*gcd(a,b) denote gcd(a, 0) = a*/
62 mt19937 rnd(time(0));
63 template <class T> T f(T x, T c, T mod) {
64     return (((uLL)x) * x % mod + c) % mod;
65 }
66 template <class T> T rho(T n) {
67     T mod = n, x = rnd() % mod, c = rnd() % (mod - 1) + 1, p = 1;
68     for (T i = 2, j = 2, d = x; ++i) {
69         x = f(x, c, mod);
70         p = ((uLL)p) * sub(x, d) % mod;
71         if (i % 127 == 0 && gcd(p, n) != 1)
72             return gcd(p, n);
73         if (i == j) {
74             j <= 1, d = x;
75             if (gcd(p, n) != 1)
76                 return gcd(p, n);
77         }
78     }
79 }
80 template <class T> T pollard_rho(T n) {
81     if (miller_rabin(n))
82         return n;
83     T p = n;
84     while (p == n)
85         p = rho(n);
86     return max(pollard_rho(p), pollard_rho(n / p));
87 }
88 int main() {
89     LL t, n, ans;
90     for (cin >> t; t--;) {
91         cin >> n;
92         ans = pollard_rho(n);
93         if (ans == n)
94             puts("Prime");
95         else
96             printf("%lld\n", ans);
97     }
98 }

```

2.11. Formula

2.11.1. Dirichlet Convolution

$$\varepsilon = \mu * 1$$

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

2.11.2. 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

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

2.11.3. Pick Theorem

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

2.11.4. Fermat's Little Theorem

$$(a + b)^p \equiv a + b \equiv a^p + b^p \pmod{p}$$

2.11.5. Wilson's Theorem

$$(p - 1)! \equiv -1 \pmod{p}$$

2.11.6. Legendre Theorem

$v(n)$:= power of p in n

$$(n)_p := \frac{n}{p^{v(n)}}$$

$s(n)$:= sum of all digits of n in base p

$$v(n!) = \sum_{i=1}^{\infty} \lfloor \frac{n}{p^i} \rfloor = \frac{n-s(n)}{p-1}$$

2.11.7. Kummer Theorem

$$v\left(\binom{n}{m}\right) = \frac{s(n) + s(m-n) - s(m)}{p-1}$$

2.11.8. ext-Kummer Theorem

$$v\left(\binom{n}{m_1, m_2, \dots, m_k}\right) = \frac{\sum_{i=1}^k s(m_i) - s(n)}{p-1}$$

2.11.9. Factorial with mod

$(n!)_p \equiv -1^{\lfloor \frac{n}{p} \rfloor} ((\lfloor \frac{n}{p} \rfloor)!)_p ((n\%p)!) \pmod{p}$ $O(p + \log_p(n))$ with factorial table.

2.11.10. Properties of nCr with mod

If any i in base p satisfies $n_i < m_i$, then $\binom{n}{m}_p \equiv 0 \pmod{p}$. Therefore

$\binom{n}{m} = \prod_{i=0}^{\max(\log_p(a), \log_p(b))} \binom{n_i}{m_i}_p \pmod{p}$ so $\binom{n}{m}_p \equiv 0 \pmod{p}$. If $p = 2$, then $\binom{n}{m}$ is odd \Leftrightarrow any bit in $n < m$. Lucas' theorem can be derived from this generating function method without relying on Fermat's Little Theorem. It is also true for polynomials.

2.11.11. ext-Lucas' Theorem

For any $k \in$ positive number, calculate $\binom{n}{m}_k$ can decompose k by Fundamental Theorem of Arithmetic. And then use crt.

2.11.12. Catalan Number

$C_0 = C_1 = 1$, if $n > 1$ then $C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k} = \frac{\binom{2n}{n}}{n+1}$. Also the number of legal placements of n pairs of brackets is C_n . If there are any k kinds of brackets available, then $k^n C_n$.

2.11.13. modinv table

$$p = i * (p/i) + p\%i, -p\%i = i * (p/i), inv(i) = -(p/i) * inv(p\%i)$$

2.12. 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 }
```

```
44 Matrix(int n, int m) {
45     v = vector<vector<T>>(n, vector<T>(m, 0));
46     in();
47 }
48 void out(int l = 0, int m = -1, int u = 0, int n = -1) {
49     if (n < 0)
50         n = v.size();
51     if (m < 0)
52         m = v[0].size();
53     for (int i = u; i < n; ++i)
54         for (int j = l; j < m; ++j)
55             printf("%lld%c", v[i][j], " \n"[j == m - 1]);
56 }
57 Matrix operator=(Matrix &b) {
58     v = b.v;
59     return *this;
60 }
61 Matrix operator+(Matrix &b) {
62     Matrix ans(*this);
63     int n = v.size(), m = v[0].size();
64     for (int i = 0; i < n; ++i)
65         for (int j = 0; j < m; ++j) {
66             ans.v[i][j] += b.v[i][j];
67             if (MOD) {
68                 if (ans.v[i][j] < 0)
69                     ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
70                 if (ans.v[i][j] >= MOD)
71                     ans.v[i][j] %= MOD;
72             }
73         }
74     return ans;
75 }
76 Matrix operator*(T x) {
77     Matrix ans(*this);
78     int n = v.size(), m = v[0].size();
79     for (int i = 0; i < n; ++i)
80         for (int j = 0; j < m; ++j) {
81             ans.v[i][j] += x;
82             if (MOD) {
83                 if (ans.v[i][j] < 0)
84                     ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
85                 if (ans.v[i][j] >= MOD)
86                     ans.v[i][j] %= MOD;
87             }
88         }
89     return ans;
90 }
91 Matrix operator-(Matrix &b) {
92     Matrix ans(*this);
93     int n = v.size(), m = v[0].size();
94     for (int i = 0; i < n; ++i)
95         for (int j = 0; j < m; ++j) {
96             ans.v[i][j] -= b.v[i][j];
97             if (MOD) {
98                 if (ans.v[i][j] < 0)
99                     ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
100                 if (ans.v[i][j] >= MOD)
101                     ans.v[i][j] %= MOD;
102             }
103         }
104     return ans;
105 }
106 Matrix operator-(T x) {
107     Matrix ans(*this);
108     int n = v.size(), m = v[0].size();
109     for (int i = 0; i < n; ++i)
110         for (int j = 0; j < m; ++j) {
111             ans.v[i][j] -= x;
112             if (MOD) {
113                 if (ans.v[i][j] < 0)
114                     ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
115                 if (ans.v[i][j] >= MOD)
116                     ans.v[i][j] %= MOD;
117             }
118         }
119     return ans;
120 }
121 Matrix operator+=(Matrix &b) {
122     int n = v.size(), m = v[0].size();
123     for (int i = 0; i < n; ++i)
124         for (int j = 0; j < m; ++j) {
125             v[i][j] += b.v[i][j];
126             if (MOD) {
127                 if (v[i][j] < 0)
128                     v[i][j] = (v[i][j] % MOD + MOD) % MOD;
129                 if (v[i][j] >= MOD)
130                     v[i][j] %= MOD;
131             }
132         }
133     return *this;
134 }
```



```

135 }
136 Matrix operator+=(T x) {
137     int n = v.size(), m = v[0].size();
138     for (int i = 0; i < n; ++i)
139         for (int j = 0; j < m; ++j) {
140             v[i][j] += x;
141             if (MOD) {
142                 if (v[i][j] < 0)
143                     v[i][j] = (v[i][j] % MOD + MOD) % MOD;
144                 if (v[i][j] >= MOD)
145                     v[i][j] %= MOD;
146             }
147         }
148     return *this;
149 }
150 Matrix operator--(Matrix &b) {
151     int n = v.size(), m = v[0].size();
152     for (int i = 0; i < n; ++i)
153         for (int j = 0; j < m; ++j) {
154             v[i][j] -= b.v[i][j];
155             if (MOD) {
156                 if (v[i][j] < 0)
157                     v[i][j] = (v[i][j] % MOD + MOD) % MOD;
158                 if (v[i][j] >= MOD)
159                     v[i][j] %= MOD;
160             }
161         }
162     return *this;
163 }
164 Matrix operator--(T x) {
165     int n = v.size(), m = v[0].size();
166     for (int i = 0; i < n; ++i)
167         for (int j = 0; j < m; ++j) {
168             v[i][j] -= x;
169             if (MOD) {
170                 if (v[i][j] < 0)
171                     v[i][j] = (v[i][j] % MOD + MOD) % MOD;
172                 if (v[i][j] >= MOD)
173                     v[i][j] %= MOD;
174             }
175         }
176     return *this;
177 }
178 Matrix operator*(Matrix &b) {
179     int n = v.size();
180     int p = b.v.size();
181     int m = b.v[0].size();
182     Matrix ans(n, m, 0);
183     for (int i = 0; i < n; ++i)
184         for (int k = 0; k < p; ++k)
185             for (int j = 0; j < m; ++j) {
186                 ans.v[i][j] += v[i][k] * b.v[k][j];
187                 if (MOD) {
188                     if (ans.v[i][j] < 0)
189                         ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
190                     if (ans.v[i][j] >= MOD)
191                         ans.v[i][j] %= MOD;
192                 }
193             }
194     return ans;
195 }
196 Matrix operator*(T x) {
197     Matrix ans(*this);
198     int n = v.size(), m = v[0].size();
199     for (int i = 0; i < n; ++i)
200         for (int j = 0; j < m; ++j) {
201             ans.v[i][j] *= x;
202             if (MOD) {
203                 if (ans.v[i][j] < 0)
204                     ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
205                 if (ans.v[i][j] >= MOD)
206                     ans.v[i][j] %= MOD;
207             }
208         }
209     return ans;
210 }
211 Matrix operator==(Matrix &b) {
212     int n = v.size();
213     int p = b.v.size();
214     int m = b.v[0].size();
215     Matrix ans(n, m, 0);
216     for (int i = 0; i < n; ++i)
217         for (int k = 0; k < p; ++k)
218             for (int j = 0; j < m; ++j) {
219                 ans.v[i][j] += v[i][k] * b.v[k][j];
220                 if (MOD) {
221                     if (ans.v[i][j] < 0)
222                         ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
223                     if (ans.v[i][j] >= MOD)
224                         ans.v[i][j] %= MOD;
225                 }
226             }
227 }
228 void gaussian() {
229     int curi = 0;
230     int n = v.size();
231     int m = v[0].size();
232     for (int j = 0; j < m; j++) {
233         int i;
234         for (i = curi; i < n; i++) {
235             if (MOD) {
236                 v[i][j] %= MOD;
237             }
238             if (v[i][j]) {
239                 break;
240             }
241         }
242         if (i >= n) {
243             continue;
244         }
245         if (v[i][j] == 0) {
246             continue;
247         }
248         for (int k = 0; k < m; k++) {
249             swap(v[i][k], v[curi][k]);
250         }
251         for (int k = m - 1; k >= j; k--) {
252             if (MOD) {
253                 v[curi][k] *= modeq(v[curi][j], (T)1, (T)MOD);
254                 v[curi][k] = (v[curi][k] % MOD + MOD) % MOD;
255             } else {
256                 v[curi][k] /= v[curi][j];
257             }
258         }
259         for (int i = 0; i < n; ++i) {
260             if (i != curi) {
261                 for (int k = m - 1; k >= j; k--) {
262                     v[i][k] -= v[curi][k] * v[i][j];
263                     if (MOD) {
264                         v[i][k] = (v[i][k] % MOD + MOD) % MOD;
265                     }
266                 }
267             }
268         }
269         curi = i;
270     }
271 }
272 void operator+=(T x) {
273     int n = v.size(), m = v[0].size();
274     for (int i = 0; i < n; ++i)
275         for (int j = 0; j < m; ++j) {
276             v[i][j] += x;
277             if (MOD) {
278                 if (v[i][j] < 0)
279                     v[i][j] = (v[i][j] % MOD + MOD) % MOD;
280                 if (v[i][j] >= MOD)
281                     v[i][j] %= MOD;
282             }
283         }
284     return *this;
285 }
286 void operator--(T x) {
287     int n = v.size(), m = v[0].size();
288     for (int i = 0; i < n; ++i)
289         for (int j = 0; j < m; ++j) {
290             v[i][j] -= x;
291             if (MOD) {
292                 if (v[i][j] < 0)
293                     v[i][j] = (v[i][j] % MOD + MOD) % MOD;
294                 if (v[i][j] >= MOD)
295                     v[i][j] %= MOD;
296             }
297         }
298     return *this;
299 }
300 void operator*(T x) {
301     Matrix ans(*this);
302     int n = v.size(), m = v[0].size();
303     for (int i = 0; i < n; ++i)
304         for (int j = 0; j < m; ++j) {
305             ans.v[i][j] *= x;
306             if (MOD) {
307                 if (ans.v[i][j] < 0)
308                     ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
309                 if (ans.v[i][j] >= MOD)
310                     ans.v[i][j] %= MOD;
311             }
312         }
313     return ans;
314 }
315 void operator*(Matrix &b) {
316     int n = v.size();
317     int p = b.v.size();
318     int m = b.v[0].size();
319     Matrix ans(n, m, 0);
320     for (int i = 0; i < n; ++i)
321         for (int k = 0; k < p; ++k)
322             for (int j = 0; j < m; ++j) {
323                 ans.v[i][j] += v[i][k] * b.v[k][j];
324                 if (MOD) {
325                     if (ans.v[i][j] < 0)
326                         ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
327                     if (ans.v[i][j] >= MOD)
328                         ans.v[i][j] %= MOD;
329                 }
330             }
331     return ans;
332 }
333 void operator==(Matrix &b) {
334     int n = v.size();
335     int p = b.v.size();
336     int m = b.v[0].size();
337     Matrix ans(n, m, 0);
338     for (int i = 0; i < n; ++i)
339         for (int k = 0; k < p; ++k)
340             for (int j = 0; j < m; ++j) {
341                 ans.v[i][j] += v[i][k] * b.v[k][j];
342                 if (MOD) {
343                     if (ans.v[i][j] < 0)
344                         ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
345                     if (ans.v[i][j] >= MOD)
346                         ans.v[i][j] %= MOD;
347                 }
348             }
349     return ans;
350 }
351 void operator/=(T x) {
352     Matrix ans(*this);
353     int n = v.size(), m = v[0].size();
354     for (int i = 0; i < n; ++i)
355         for (int j = 0; j < m; ++j) {
356             ans.v[i][j] /= x;
357             if (MOD) {
358                 ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
359                 if (ans.v[i][j] >= MOD)
360                     ans.v[i][j] %= MOD;
361             } else {
362                 ans.v[i][j] /= x;
363             }
364         }
365     return ans;
366 }
367 void operator/=(Matrix &b) {
368     Matrix ans(*this);
369     int n = v.size(), m = v[0].size();
370     for (int i = 0; i < n; ++i)
371         for (int j = 0; j < m; ++j) {
372             ans.v[i][j] /= b.v[i][j];
373             if (MOD) {
374                 ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
375                 if (ans.v[i][j] >= MOD)
376                     ans.v[i][j] %= MOD;
377             } else {
378                 ans.v[i][j] /= b.v[i][j];
379             }
380         }
381     return ans;
382 }
383 void operator/=(T x) {
384     Matrix ans(*this);
385     int n = v.size(), m = v[0].size();
386     for (int i = 0; i < n; ++i)
387         for (int j = 0; j < m; ++j) {
388             ans.v[i][j] /= x;
389             if (MOD) {
390                 ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
391                 if (ans.v[i][j] >= MOD)
392                     ans.v[i][j] %= MOD;
393             } else {
394                 ans.v[i][j] /= x;
395             }
396         }
397     return ans;
398 }
399 void operator/=(Matrix &b) {
400     Matrix ans(*this);
401     int n = v.size(), m = v[0].size();
402     for (int i = 0; i < n; ++i)
403         for (int j = 0; j < m; ++j) {
404             ans.v[i][j] /= b.v[i][j];
405             if (MOD) {
406                 ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
407                 if (ans.v[i][j] >= MOD)
408                     ans.v[i][j] %= MOD;
409             } else {
410                 ans.v[i][j] /= b.v[i][j];
411             }
412         }
413     return ans;
414 }

```

```

315         v[i][k] -= v[curi][k] * v[i][j];
316         if (MOD) {
317             v[i][k] = (v[i][k] % MOD + MOD) % MOD;
318         }
319     }
320 }
321 curi++;
322 }
323 };

```

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];

```

```

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 }

```

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[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++;

```



```

cycle[x] = cnt;
r[x] = 0;
len[cnt] = 1;
int temp = a[x];
while (temp ^ x) {
    r[temp] = len[cnt];
    len[cnt]++;
    cycle[temp] = cnt;
    temp = a[temp];
}
}

void dfs(int x) {
    if (vis2[x])
        return;
    vis2[x] = 1;
    for (int i : v[x]) {
        dfs(i);
    }
}

void dfs2(int x) {
    if (cycle[x] || d[x])
        return;
    dfs2(a[x]);
    d[x] = d[a[x]] + 1;
    r[x] = r[a[x]];
    cycle[x] = cycle[a[x]];
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        v[i].push_back(a[i]);
        v[a[i]].push_back(i);
    }
    for (int i = 1; i <= n; i++) {
        if (!vis2[i]) {
            findcycle(i);
            dfs(i);
        }
    }
    for (int i = 1; i <= n; i++) {
        if (!cycle[i] && !r[i]) {
            dfs2(i);
        }
    }
    for (int i = 1; i <= n; i++) {
        cout << d[i] + len[cycle[i]] << " ";
    }
}

```

4.2. Dijkstra

```

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

5 void dijkstra(int x) {
    priority_queue<pair<int, int>, vector<pair<int, int>>,
        greater<pair<int, int>>>
        pq;
    memset(dis, 0x3f, sizeof(dis));
    dis[x] = 0;
    pq.push({0, x});
    while (!pq.empty()) {
        pair<int, int> now = pq.top();
        pq.pop();
        if (vis[now.second])
            continue;
        vis[now.second] = 1;
        for (auto [i, w] : v[now.second]) {
            if (vis[i])
                continue;
            if (dis[now.second] + w < dis[i]) {
                dis[i] = dis[now.second] + w;
                pq.push({dis[i], i});
            }
        }
    }
}

```

4.3. MaximumFlow

```

1 #define int long long

```

```

3 using namespace std;

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

11 bool bfs() {
13     memset(head, 0, sizeof(head));
    memset(lv, 0, sizeof(lv));
15     queue<int> q;
    q.push(1);
17     while (!q.empty()) {
        int now = q.front();
        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);
            }
        }
    }
29     return lv[n];
}

31 int dfs(int x, int flow) {
33     int ret = 0;
    if (x == n)
        return flow;
35     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;
        }
    }
47     return ret;
}

49 signed main() {
51     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;
    }
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])
            continue;
        if (d[i]) {
            low[x] = min(low[x], d[i]);
        } else {
            dfs(i);
            low[x] = min(low[x], low[i]);
        }
    }
23     if (d[x] == low[x]) {

```

```

    now++;
    while (!s.empty()) {
        int k = s.top();
        s.pop();
        scc[k] = now;
        if (k == x)
            break;
    }
}
}

```

4.5. 2-SAT(CSES Giant Pizza)

```

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

```

```

    }
    for (int i = 1; i <= 2 * m; i++) {
        for (int j : v[i]) {
            if (scc[i] ^ scc[j]) {
                v2[scc[j]].push_back(scc[i]);
                ind[scc[i]]++;
            }
        }
    }
    for (int i = 1; i <= now; i++) {
        if (!ind[i]) {
            q.push(i);
        }
    }
    while (!q.empty()) {
        int k = q.front();
        q.pop();
        if (!ans[k]) {
            ans[k] = 1;
            ans[op[k]] = 2;
        }
        for (int i : v2[k]) {
            ind[i]--;
            if (!ind[i]) {
                q.push(i);
            }
        }
    }
    for (int i = 1; i <= m; i++) {
        if (ans[scc[i]] == 1) {
            cout << "+ ";
        } else {
            cout << "- ";
        }
    }
}

```

5. DP

5.1. Li-Chao Segment Tree

```

1 struct line {
2     int a, b = 1000000000000000;
3     int y(int x) { return a * x + b; }
4 };
5
6 line tree[4000005];
7 int n, x;
8 int s[200005];
9 int f[200005];
10 int dp[200005];
11
12 void update(line ins, int l = 1, int r = 1e6, int index = 1) {
13     if (l == r) {
14         if (ins.y(l) < tree[index].y(l)) {
15             tree[index] = ins;
16         }
17         return;
18     }
19     int mid = (l + r) >> 1;
20     if (tree[index].a < ins.a)
21         swap(tree[index], ins);
22     if (tree[index].y(mid) > ins.y(mid)) {
23         swap(tree[index], ins);
24         update(ins, l, mid, index << 1);
25     } else {
26         update(ins, mid + 1, r, index << 1 | 1);
27     }
28 }
29
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
struct CHO {
7   deque<line> dq;
   int intersect(line x, line y) {
9       int d1 = x.b - y.b;
       int d2 = y.a - x.a;
11      return d1 / d2;
   }
13   bool check(line x, line y, line z) {
       int I12 = intersect(x, y);
       int I23 = intersect(y, z);
15      return I12 < I23;
   }
17   void insert(int a, int b) {
       if (!dq.empty() && a == dq.back().a)
19         return;
       while (dq.size() >= 2 &&
21             !check(dq[dq.size() - 2], dq[dq.size() - 1], {a, b})) {
23         dq.pop_back();
       }
25      dq.push_back({a, b});
   }
27   void update(int x) {
       while (dq.size() >= 2 && dq[0].y(x) >= dq[1].y(x)) {
29         dq.pop_front();
       }
31   }
   int query(int x) {
33       update(x);
       return dq.front().y(x);
35   }
};

```

6. Geometry

6.1. Intersect

```

1 struct point {
   int x, y;
3   point operator+(point b) { return {x + b.x, y + b.y}; }
   point operator-(point b) { return {x - b.x, y - b.y}; }
5   int operator*(point b) { return x * b.x + y * b.y; }
   int operator^(point b) { return x * b.y - y * b.x; }
7 };
9 bool onseg(point x, point y, point z) {
   return ((x - z) ^ (y - z)) == 0 && (x - z) * (y - z) <= 0;
11 }
13 int dir(point x, point y) {
   int k = x ^ y;
15   if (k == 0)
       return 0;
   if (k > 0)
17     return 1;
   if (k < 0)
       return -1;
19 }
21 bool intersect(point x, point y, point z, point w) {
23   if (onseg(x, y, z) || onseg(x, y, w))
       return 1;
   if (onseg(z, w, x) || onseg(z, w, y))
25     return 1;
   if (dir(y - x, z - x) * dir(y - x, w - x) == -1 &&
27       dir(z - w, x - w) * dir(z - w, y - w) == -1) {
       return 1;
29   }
   return 0;
31 }

```

6.2. Inside

```

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

```

6.3. Minimum Euclidean Distance

```

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

```

6.4. Convex Hull

```

1 #define int long long
2 #define fastio
3 ios_base::sync_with_stdio(0);
4 cin.tie(0);
5 cout.tie(0);
6
7 using namespace std;
8
9 template <typename T> pair<T, T> operator-(pair<T, T> a, pair<T, T> b) {
10     return make_pair(a.first - b.first, a.second - b.second);
11 }
12
13 template <typename T> T cross(pair<T, T> a, pair<T, T> b) {
14     return a.first * b.second - a.second * b.first;
15 }
16
17 template <typename T> vector<pair<T, T>> getCH(vector<pair<T, T>> v) {
18     int n = v.size();
19     sort(v.begin(), v.end());
20     vector<pair<T, T>> hull;
21     for (int i = 0; i < 2; i++) {
22         int t = hull.size();
23         for (auto x : v) {
24             while (hull.size() - t >= 2 &&
25                   cross(hull[hull.size() - 1] - hull[hull.size() - 2],
26                       x - hull[hull.size() - 2]) <= 0)
27                 hull.pop_back();
28             hull.push_back(x);
29         }
30         hull.pop_back();
31         reverse(v.begin(), v.end());
32     }
33     return hull;
34 }

```

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;
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);
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         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
4 int n, q;
5 int a[200005][21];
6 int d[200005];
7 vector<int> v[200005];
8
9 void init() {
10     for (int j = 1; j < 21; j++) {
11         for (int i = 1; i <= n; i++) {
12             a[i][j] = a[a[i][j - 1]][j - 1];
13         }
14     }
15 }
16
17 void dfs(int x, int pre) {
18     for (int i : v[x]) {
19         if (i == pre)
20             continue;
21         a[i][0] = x;
22         d[i] = d[x] + 1;
23         dfs(i, x);
24     }
25 }
26
27 int lca(int x, int y) {
28     while (d[x] ^ d[y]) {
29         if (d[x] < d[y]) {
30             swap(x, y);
31         }
32         int k = __lg(d[x] - d[y]);
33         x = a[x][k];
34     }
35     if (x == y) {
36         return x;
37     }
38     for (int i = 20; i >= 0; i--) {
39         if (a[x][i] != a[y][i]) {
40             x = a[x][i];
41             y = a[y][i];
42         }
43     }

```

```

    }
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     }
}

```

8. Misc

8.1. Tri Search

```

1  using namespace std;
3  int n;
   double a[15], x, y;
5
   double get(double x) {
7       double ret = 0;
       double k = 1;
9       for (int i = 0; i <= n; i++) {
           ret += k * a[i];
11          k *= x;
       }
13       return -ret;
   }
15
   template <class T> T bi_search(T l, T r, T end) {
17       if (!check(r - end))
           return r - end;
19       for (; r - l > end;) {
           T mid = (l + r) / 2;
21           if (check(mid))
               r = mid;
23           else
               l = mid;
25       }
       return l;
27   }
   /*check gives 0000000111 find the last 0*/
29
   template <class T> T tri_search(T l, T r, T end) {
31       T midl, midr;
       for (;;) {
33           midl = (l + r) / 2;
           midr = (midl + r) / 2;
35           if (midr - midl < end)
               break;
37           if (get(midr) > get(midl))
               r = midr;
39           else
               l = midl;
41       }
       for (; r - l > end;) {
43           midl = (l + r) / 2;
           if (get(r) > get(l))
45               r = midl;
           else
47               l = midl;
       }
49       return l;
   }
51 /*get gives the value, find the minimum*/

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

```