

## Contents

### 1 DataStructure

1.1	Treap	1
1.2	Dynamic Segment Tree	1
1.3	Pbds Gp Hash Table	2
1.4	Pbds Rope	2
1.5	Pbds Tree	2
1.6	Pbds Priority Queue	2
1.7	2DBIT	2
1.8	Persistent Segment Tree	3

### 2 Math

2.1	FFT	3
2.2	NTT	3
2.3	FWT	3
2.4	Gaussian-Jordan	4
2.5	Mu	4
2.6	Lucas	4
2.7	Inv	4
2.8	CRT	4
2.9	Generator	4
2.10	Count Primes	5
2.11	Pollard Rho	6
2.12	Formula	6
2.12.1	Dirichlet Convolution	6
2.12.2	Burnside's Lemma	6
2.12.3	Pick Theorem	6
2.12.4	Fermat's Little Theorem	6
2.12.5	Wilson's Theorem	6
2.12.6	Legendre Theorem	6
2.12.7	Kummer Theorem	6
2.12.8	ext-Kummer Theorem	6
2.12.9	Factorial with mod	7
2.12.10	Properties of nCr with mod	7
2.12.11	ext-Lucas' Theorem	7
2.12.12	Catalan Number	7
2.12.13	modinv table	7
2.13	Matrix	7
2.14	Miller Rabin	8
2.15	Xor Basis	9

### 3 String

3.1	KMP	9
3.2	Longest Palindrome	9
3.3	Z	9
3.4	Booth	10

### 4 Graph

4.1	one-out-degree (CSES Planets Cycles)	10
4.2	Dijkstra	10
4.3	MaximumFlow	10
4.4	Dinic	11
4.5	SCC	11
4.6	VBCC	11
4.7	2-SAT(CSES Giant Pizza)	12

### 5 DP

5.1	Li-Chao Segment Tree	12
5.2	CHO	13
5.3	SOSDP	13

### 6 Geometry

6.1	Intersect	13
6.2	Inside	13
6.3	Minimum Euclidean Distance	13
6.4	Convex Hull	13
6.5	164253 Version	14

### 7 Tree

7.1	Heavy Light Decomposition (modify and query on path)	14
7.2	LCA	15

### 8 Misc

8.1	Tri Search	15
8.2	Big Number	16
8.3	對拍	16

## 1. DataStructure

### 1.1. Treap

```

1 #define pii pair<int, int>
2 struct node {
3     int tag = 0;
4     int sum = 0;
5     int prio = rand();
6     int lson = 0;
7     int rson = 0;
8     int si = 0;
9     int val = 0;
10 };
11 node treap[400005];
12 int cnt = 0;
13 int root = 0;
14
15 void update(int index) {
16     int lson = treap[index].lson;
17     int rson = treap[index].rson;
18     treap[index].si = treap[lson].si + treap[rson].si + 1;
19     treap[index].sum = treap[lson].sum;
20     treap[index].sum += treap[rson].sum;
21     treap[index].sum += treap[index].val;
22 }
23 void push(int index) {
24     if (!treap[index].tag)
25         return;
26     swap(treap[index].lson, treap[index].rson);
27     int lson = treap[index].lson;
28     int rson = treap[index].rson;
29     treap[lson].tag ^= 1;
30     treap[rson].tag ^= 1;
31     treap[index].tag = 0;
32 }
33 pii split(int rk, int index) {
34     if (!index)
35         return {0, 0};
36     push(index);
37     int lson = treap[index].lson;
38     int rson = treap[index].rson;
39     if (rk <= treap[lson].si) {
40         pii temp = split(rk, lson);
41         treap[index].lson = temp.second;
42         update(index);
43         return {temp.first, index};
44     } else {
45         pii temp = split(rk - treap[lson].si - 1, rson);
46         treap[index].rson = temp.first;
47         update(index);
48         return {index, temp.second};
49     }
50 }
51
52 int merge(int x, int y) {
53     if (!x && !y)
54         return 0;
55     if (!x && y)
56         return y;
57     if (x && !y)
58         return x;
59     push(x);
60     push(y);
61     if (treap[x].prio < treap[y].prio) {
62         treap[x].rson = merge(treap[x].rson, y);
63         update(x);
64         return x;
65     } else {
66         treap[y].lson = merge(x, treap[y].lson);
67         update(y);
68         return y;
69     }
70 }
71
72 void insert(int x, int v) {
73     pii temp = split(x - 1, root);
74     cnt++;
75     treap[cnt].val = v;
76     update(cnt);
77     temp.first = merge(temp.first, cnt);
78     root = merge(temp.first, temp.second);
79 }
80
81 int query(int l, int r) {
82     pii R = split(r, root);
83     pii L = split(l - 1, R.first);
84     int ret = treap[L.second].sum;
85     R.first = merge(L.first, L.second);
86     root = merge(R.first, R.second);
87     return ret;
88 }

```

```

91 void modify(int l, int r) {
    pii R = split(r, root);
93     pii L = split(l - 1, R.first);
    treap[L.second].tag ^= 1;
95     R.first = merge(L.first, L.second);
    root = merge(R.first, R.second);
97 }

```

## 1.2. Dynamic Segment Tree

```

1  #define int long long
3  using namespace std;

5  int n, q;
   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) {
        tree[index].lson = ++now;
19     }
     if (!tree[index].rson) {
21         tree[index].rson = ++now;
     }
23     int lson = tree[index].lson;
     int rson = tree[index].rson;
25     tree[lson].tag += tree[index].tag;
     tree[rson].tag += tree[index].tag;
27     tree[index].data = tree[index].rv();
     tree[index].tag = 0;
29 }

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

51 int query(int l, int r, int L, int R, int index) {
   // cout << L << " " << R << "\n";
53     if (l == L && r == R) {
        return tree[index].rv();
55     }
     int mid = (l + r) >> 1;
     push(index);
57     int lson = tree[index].lson;
     int rson = tree[index].rson;
59     if (R <= mid) {
        return query(l, mid, L, R, lson);
61     }
     if (L > mid) {
63         return query(mid + 1, r, L, R, rson);
65     }
     return query(l, mid, L, mid, lson) + query(mid + 1, r, mid + 1, R, rson);
67 }

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

```

```

83         cin >> x >> y;
        cout << query(1, mx, x, y, 1) << "\n";
85     } else {
        cin >> x >> y;
87         modify(1, mx, a[x], a[x], -1, 1);
        a[x] = y;
89         modify(1, mx, a[x], a[x], 1, 1);
    }
91 }

```

## 1.3. Pbds Gp Hash Table

```

1  using namespace __gnu_pbds;
3  #define ull unsigned ll
   mt19937 mt(hash<string>("164253_official_beautiful_fruit"));
5  struct myhash {
       static ull splitmix64(ull x) {
7           x += 0x9e3779b97f4a7c15;
           x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
9           x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
           return x ^ (x >> 31);
11        }
       ull operator()(ull x) const {
13           static const ull FIXED_RANDOM =
               (ull)make_unique<char>().get() ^
               chrono::high_resolution_clock::now().time_since_epoch().count();
15           // static const ull FIXED_RANDOM=mt();
           // static const ull
           // FIXED_RANDOM=chrono::steady_clock::now().time_since_epoch().count();
17           return splitmix64(x + FIXED_RANDOM);
19       }
21 };
   /*
23   gp_hash_table<ull,ull,myhash> gp;
   gp[x]=y;
25   if(gp.find(x)!=gp.end())cout<<gp[x];
   gp.count(); //CE
27   */

```

## 1.4. Pbds Rope

```

1  using namespace __gnu_cxx;
3  /*
   rope<int> r;
5   r.erase(pos,k); //r=r.[0,pos)+r.[pos+k,r.length());
   push_back(x) pop_back() insert(pos,x) clear() find(x) lower_bound(a)
7   upper_bound //same as vector r.length(); //same as .length
   r.replace(pos,len=r.length(),x); //r.[pos,pos+len)=x;
9   r.substr(pos,x); //return r.[pos,pos+x);
   rope<char> s="official_beautiful_fruit";
11  cout<<s; //it's legal
   */

```

## 1.5. Pbds Tree

```

1  using namespace __gnu_pbds;
3  /*
   tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_update> tr;
5   //same as rope<int>, except tr.lower_bound(x) and upper_bound
   tr.find_by_order(k); //return kth iterator; k=[0,tr.size()), out of
7   get tr.end() tr.order_of_key(val); //return rank(val); tr.join(tr2)
   and tr2, tr2.clear() tr.split(const int&r, RBTree&tr2); //<r will in tr2
9   in tr2
   */

```

## 1.6. Pbds Priority Queue

```

1  __gnu_pbds::priority_queue<int> pq;
3  /*
   push(x); //return iterator
5   pop() top() join(pq2) erase(iter) modify(iter,x)
   */

```

## 1.7. 2DBIT

```

1  using namespace std;
   #define LL long long
5   #define pii pair<int, int>
   #define N 1005
7   #define F first
   #define S second
9   #define bit[N][N];
   #define lb(x) (x & -x)
11 void upd(int i, int j, int v) {
   for (; j < N; j += lb(j))

```

```

13     for (int k = i; k < N; k += lb(k))
14         bit[k][j] += v;
15 }
16 int qry2(int i, int j) {
17     int ans = 0;
18     for (; j; j -= lb(j))
19         for (int k = i; k; k -= lb(k))
20             ans += bit[k][j];
21     return ans;
22 }
23 int qry(int y1, int x1, int y2, int x2) {
24     return qry2(y2, x2) - qry2(y2, x1 - 1) - qry2(y1 - 1, x2) +
25         qry2(y1 - 1, x1 - 1);
26 }
27 int main() {
28     int n, q, i = 1, j, y, x;
29     for (scanf("%d %d", &n, &q); getchar(), i <= n; ++i)
30         for (j = 1; j <= n; ++j)
31             if (getchar() == '*')
32                 upd(i, j, 1);
33     for (; q--;) {
34         scanf("%d", &i);
35         if (i == 1)
36             scanf("%d%d", &i, &j), upd(i, j, 1 - 2 * qry(i, j, i, j));
37         else
38             scanf("%d%d%d%d", &i, &j, &y, &x), printf("%d\n", qry(i, j, y, x));
39     }
40 }

```

## 1.8. Persistent Segment Tree

```

1 // cses Range Queries and Copies
2
3 using namespace std;
4 #define LL long long
5 #define pii pair<int, int>
6 #define N 200005
7 #define F first
8 #define S second
9 int n, ver = 1;
10 LL a[N];
11 struct Seg {
12     LL v = 0;
13     struct Seg *l = NULL, *r = NULL;
14 }
15 #define M (L + R >> 1)
16 static const void init(Seg *node, int L = 1, int R = n) {
17     if (L == R) {
18         node->v = a[L];
19         return;
20     }
21     node->l = new Seg();
22     init(node->l, L, M);
23     node->r = new Seg();
24     init(node->r, M + 1, R);
25     node->v = node->l->v + node->r->v;
26 }
27 static const void upd(Seg *node, int x, LL v, int L = 1, int R = n) {
28     if (L == R) {
29         node->v = v;
30         return;
31     }
32     if (x <= M)
33         node->l = new Seg(*node->l), upd(node->l, x, v, L, M);
34     else
35         node->r = new Seg(*node->r), upd(node->r, x, v, M + 1, R);
36     node->v = node->l->v + node->r->v;
37 }
38 static const LL qry(Seg *node, int l, int r, int L = 1, int R = n) {
39     if (l <= L && R <= r)
40         return node->v;
41     if (r <= M)
42         return qry(node->l, l, r, L, M);
43     if (M + 1 <= l)
44         return qry(node->r, l, r, M + 1, R);
45     return qry(node->l, l, M, L, M) + qry(node->r, M + 1, r, M + 1, R);
46 }
47 * tree[N];
48 int main() {
49     ios::sync_with_stdio(0);
50     cin.tie(0);
51     cout.tie(0);
52     int q, i = 1, j, k;
53     for (cin >> n >> q; i <= n; ++i)
54         cin >> a[i];
55     tree[1] = new Seg();
56     Seg::init(tree[1]);
57     for (; q--;) {
58         cin >> i >> k;
59         if (i == 1)
60             cin >> i >> j, Seg::upd(tree[k], i, j);
61         else if (i == 2)
62             cin >> i >> j, cout << Seg::qry(tree[k], i, j) << "\n";
63         else

```

```

63         tree[++ver] = new Seg(*tree[k]);
64     }
65 }

```

## 2. Math

### 2.1. FFT

```

1 using namespace std;
2 inline int read() {
3     int ans = 0;
4     char c = getchar();
5     while (!isdigit(c))
6         c = getchar();
7     while (isdigit(c)) {
8         ans = ans * 10 + c - '0';
9         c = getchar();
10    }
11    return ans;
12 }
13 typedef complex<double> comp;
14 const int MAXN = 1000005;
15 const comp I(0, 1);
16 const double PI = acos(-1);
17 comp A[MAXN * 3], B[MAXN * 3], tmp[MAXN * 3], ans[MAXN * 3];
18 void fft(comp F[], int N, int sgn = 1) {
19     if (N == 1)
20         return;
21     memcpy(tmp, F, sizeof(comp) * N);
22     for (int i = 0; i < N; i++)
23         *(i % 2 ? F + i / 2 + N / 2 : F + i / 2) = tmp[i];
24     fft(F, N / 2, sgn), fft(F + N / 2, N / 2, sgn);
25     comp *G = F, *H = F + N / 2;
26     comp cur = 1, step = exp(2 * PI / N * sgn * I);
27     for (int k = 0; k < N / 2; k++) {
28         tmp[k] = G[k] + cur * H[k];
29         tmp[k + N / 2] = G[k] - cur * H[k];
30         cur *= step;
31     }
32     memcpy(F, tmp, sizeof(comp) * N);
33 }
34 int main() {
35     int n = read(), m = read(), N = 1 << __lg(n + m + 1) + 1;
36     for (int i = 0; i <= n; ++i)
37         A[i] = read();
38     for (int i = 0; i <= m; ++i)
39         B[i] = read();
40     fft(A, N), fft(B, N);
41     for (int i = 0; i < N; ++i)
42         ans[i] = A[i] * B[i];
43     fft(ans, N, -1);
44     for (int i = 0; i <= n + m; ++i)
45         printf("%d ", int(ans[i].real() / N + 0.1));
46     return 0;
47 }

```

### 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        x %= MOD;
14        y >>= 1;
15    }
16    return ret;
17 }
18
19 void ntt(int F[], int N, int sgn) {
20     int bit = __lg(N);
21     for (int i = 0; i < N; ++i) {
22         rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
23         if (i < rev[i])
24             swap(F[i], F[rev[i]]);
25     }
26     for (int l = 1, t = 1; l < N; l <= 1, t++) {
27         int step = qpow(G, ((MOD - 1) >> t) * sgn + MOD - 1);
28         for (int i = 0; i < N; i += l << 1)

```

```

33     for (int k = i, cur = 1; k < i + l; ++k) {
34         int g = F[k], h = (ll)F[k + l] * cur % MOD;
35         F[k] = (g + h) % MOD;
36         F[k + l] = ((g - h) % MOD + MOD) % MOD;
37         cur = (ll)cur * step % MOD;
38     }
39 }
40 if (sgn == -1) {
41     int invN = qpow(N, MOD - 2);
42     for (int i = 0; i < N; ++i)
43         F[i] = (ll)F[i] * invN % MOD;
44 }
45 }

```

### 2.3. FWT

```

1 #define LOGN 21
2 #define N (1 << LOGN)
3 void fwt(ll f[], int rev) {
4     for (int k = 1; k < LOGN; ++k) {
5         for (int i = 0, m = 1 << k - 1; i + m < N; i += 1 << k) {
6             for (int j = 0; j < m; ++j) {
7                 ll u = f[i + j], v = f[i + j + m];
8                 f[i + j] = u + v;
9                 f[i + j + m] = u - v;
10                if (rev)
11                    f[i + j] >>= 1, f[i + j + m] >>= 1;
12            }
13        }
14    }
15 }

```

### 2.4. Gaussian-Jordan

```

1 #define int long long
2 using namespace std;
3
4 int n;
5 double a[105][105];
6
7 // n <= m
8 void gaussian(double a[105][105], int n, int m) {
9     int curi = 0;
10    for (int j = 0; j < m; j++) {
11        int i;
12        for (i = curi; i < n; i++) {
13            if (a[i][j]) {
14                break;
15            }
16        }
17        if (a[i][j] == 0)
18            continue;
19        for (int k = 0; k < m; k++) {
20            swap(a[i][k], a[curi][k]);
21        }
22        for (int k = m - 1; k >= j; k--) {
23            a[curi][k] /= a[curi][j];
24        }
25        for (int i = 0; i < n; ++i) {
26            if (i != curi) {
27                for (int k = m - 1; k >= j; k--) {
28                    a[i][k] -= a[curi][k] * a[i][j];
29                }
30            }
31        }
32        curi++;
33    }
34 }
35 }

```

### 2.5. Mu

```

1 vector<int> prime;
2 bitset<1000005> vis;
3 int n;
4 int mu[1000005];
5
6 void init() {
7     for (int i = 2; i <= n; i++) {
8         if (!vis[i]) {
9             prime.push_back(i);
10            mu[i] = -1;
11        }
12        for (int p : prime) {
13            if (i * p > n)
14                break;
15            vis[i * p] = 1;
16            if (i % p == 0) {
17                mu[i * p] = 0;
18                break;
19            } else {
20                mu[i * p] = mu[i] * mu[p];
21            }
22        }
23    }
24 }

```

```

21     }
22 }
23 }

```

### 2.6. Lucas

```

1 int fact[100005];
2 int p;
3
4 void init() {
5     fact[0] = 1;
6     for (int i = 1; i <= p; i++) {
7         fact[i] = fact[i - 1] * i % p;
8     }
9 }
10
11 int inv(int x, int p) {
12     if (x == 1)
13         return 1;
14     return (p - p / x) * inv(p % x, p) % p;
15 }
16
17 int c(int x, int y, int p) {
18     if (x < y)
19         return 0;
20     int k = fact[x] * inv(fact[y], p) % p;
21     return k * inv(fact[x - y], p) % p;
22 }
23
24 int lucas(int x, int y, int p) {
25     if (x == 0)
26         return 1;
27     return lucas(x / p, y / p, p) % p * c(x % p, y % p, p) % p;
28 }

```

### 2.7. Inv

```

1 int exgcd(int a, int b, int &x, int &y) {
2     if (b == 0) {
3         x = 1;
4         y = 0;
5         return a;
6     }
7     int d = exgcd(b, a % b, y, x);
8     y -= x * (a / b);
9     return d;
10 }
11
12 int inv(int a, int p) {
13     int x, y;
14     exgcd(a, p, x, y);
15     return (x % p + p) % p;
16 }

```

### 2.8. CRT

```

1 #define int long long
2 using namespace std;
3
4 int n;
5 int a[15];
6 int b[15];
7 int mul = 1;
8
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
19 int inv(int a, int p) {
20     int x, y;
21     exgcd(a, p, x, y);
22     return x;
23 }
24
25 int ans = 0;
26
27 signed main() {
28     cin >> n;
29     for (int i = 1; i <= n; i++) {
30         cin >> a[i] >> b[i];
31         mul *= a[i];
32     }
33     for (int i = 1; i <= n; i++) {
34         ans += inv(mul / a[i], a[i]) * (mul / a[i]) % mul * b[i] % mul;
35     }
36 }

```

```

    ans %= mul;
37 }
    ans = (ans + mul) % mul;
39 cout << ans;
}

```

## 2.9. Generator

```

1 #define int long long
3 using namespace std;

5 int t;
7 int n, d;
7 bitset<1000005> exist;
7 bitset<1000005> vis;
9 vector<int> prime;
9 int phi[1000005];

11 void init() {
13     phi[1] = 1;
15     for (int i = 2; i <= 1000000; i++) {
17         if (!vis[i]) {
19             prime.push_back(i);
21             phi[i] = i - 1;
23             for (int j : prime) {
25                 if (i * j > 1000000)
27                     break;
29                 vis[i * j] = 1;
31                 if (i % j == 0) {
33                     phi[i * j] = phi[i] * j;
35                     break;
37                 } else {
39                     phi[i * j] = phi[i] * phi[j];
41                 }
43             }
45             exist[2] = exist[4] = 1;
47             for (int i : prime) {
49                 if (i == 2)
51                     continue;
53                 for (int j = i; j <= 1000000; j *= i) {
55                     exist[j] = 1;
57                     if (j * 2 <= 1000000) {
59                         exist[j * 2] = 1;
61                     }
63                 }
65             }
67         }
69     }

71 vector<int> factors(int x) {
73     vector<int> v;
75     for (int i = 1; i * i <= x; i++) {
77         if (x % i == 0) {
79             v.push_back(i);
81             if (i * i != x) {
83                 v.push_back(x / i);
85             }
87         }
89     }
91     return v;
93 }

95 int f(int x, int y, int mod) {
97     int ret = 1;
99     while (y) {
101         if (y & 1) {
103             ret *= x;
105             ret %= mod;
107         }
109         x *= x;
111         x %= mod;
113         y >>= 1;
115     }
117     return (ret % mod + mod) % mod;
119 }

121 vector<int> findroot(int x) {
123     vector<int> ret;
125     if (!exist[x])
127         return ret;
129     int phix = phi[x];
131     vector<int> fact = factors(phix);
133     int fst;
135     for (int i = 1; i <= phix; i++) {
137         if (__gcd(i, x) != 1)
139             continue;
141         bool ok = 1;
143         for (int j : fact) {
145             if (j != phix && f(i, j, x) == 1) {
147                 ok = 0;
149                 break;
151             }
153         }
155         if (ok) {
157             ret.push_back(i);
159             break;
161         }
163     }
165     return ret;
167 }

169 signed main() {
171     ios::sync_with_stdio(0);
173     cin.tie(0);
175     cout.tie(0);
177     init();
179     cin >> t;
181     while (t--) {
183         cin >> n >> d;
185         vector<int> v = findroot(n);
187         sort(v.begin(), v.end());
189         cout << v.size() << "\n";
191         for (int i = 0; i < v.size(); i++) {
193             if (i % d == d - 1) {
195                 cout << v[i] << " ";
197             }
199         }
201         cout << "\n";
203     }
205 }

```

```

87 }
89 if (ok) {
91     fst = i;
93     break;
95 }
97 int now = fst;
99 // cout << fst << "\n";
101 for (int i = 1; i <= phix; i++) {
103     if (__gcd(i, phix) == 1) {
105         ret.push_back(now);
107     }
109     now *= fst;
111     now %= x;
113 }
115 return ret;
117 }

119 signed main() {
121     ios::sync_with_stdio(0);
123     cin.tie(0);
125     cout.tie(0);
127     init();
129     cin >> t;
131     while (t--) {
133         cin >> n >> d;
135         vector<int> v = findroot(n);
137         sort(v.begin(), v.end());
139         cout << v.size() << "\n";
141         for (int i = 0; i < v.size(); i++) {
143             if (i % d == d - 1) {
145                 cout << v[i] << " ";
147             }
149         }
151         cout << "\n";
153     }
155 }

```

## 2.10. Count Primes

```

1 using namespace std;
3 using i64 = long long;
5 i64 count_pi(i64 N) {
7     if (N <= 1)
9         return 0;
11     int v = sqrt(N + 0.5);
13     int n_4 = sqrt(v + 0.5);
15     int T = min((int)sqrt(n_4) * 2, n_4);
17     int K = pow(N, 0.625) / log(N) * 2;
19     K = max(K, v);
21     K = min<i64>(K, N);
23     int B = N / K;
25     B = N / (N / B);
27     B = min<i64>(N / (N / B), K);

29 vector<i64> l(v + 1);
31 vector<int> s(K + 1);
33 vector<bool> e(K + 1);
35 vector<int> w(K + 1);
37 for (int i = 1; i <= v; ++i)
39     l[i] = N / i - 1;
41 for (int i = 1; i <= v; ++i)
43     s[i] = i - 1;

45 const auto div = [](i64 n, int d) -> int { return double(n) / d; };
47 int p;
49 for (p = 2; p <= T; ++p)
51     if (s[p] != s[p - 1]) {
53         i64 M = N / p;
55         int t = v / p, t0 = s[p - 1];
57         for (int i = 1; i <= t; ++i)
59             l[i] -= l[i * p] - t0;
61         for (int i = t + 1; i <= v; ++i)
63             l[i] -= s[div(M, i)] - t0;
65         for (int i = v, j = t; j >= p; --j)
67             for (int l = j * p; i >= l; --i)
69                 s[i] -= s[j] - t0;
71         for (int i = p * p; i <= K; i += p)
73             e[i] = 1;
75     }

77 e[1] = 1;
79 int cnt = 1;
81 vector<int> roughs(B + 1);
83 for (int i = 1; i <= B; ++i)
85     if (!e[i])
87         roughs[cnt++] = i;
89 roughs[cnt] = 0x7fffffff;
91 for (int i = 1; i <= K; ++i)
93     w[i] = e[i] + w[i - 1];
95 for (int i = 1; i <= K; ++i)
97     s[i] = w[i] - w[i - (i & -i)];

```

```

53 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 const auto add = [&](int x) -> void {
61     e[x] = 1;
62     while (x <= K)
63         ++s[x], x += x & -x;
64 };
65 cnt = 1;
66 for (; p <= n_4; ++p)
67     if (!e[p]) {
68         i64 q = i64(p) * p, M = N / p;
69         while (cnt < q)
70             w[cnt] = query(cnt), cnt++;
71         int t1 = B / p, t2 = min<i64>(B, M / q), t0 = query(p - 1);
72         int id = 1, i = 1;
73         for (; i <= t1; i = roughs[++id])
74             l[i] -= l[i * p] - t0;
75         for (; i <= t2; i = roughs[++id])
76             l[i] -= query(div(M, i)) - t0;
77         for (; i <= B; i = roughs[++id])
78             l[i] -= w[div(M, i)] - t0;
79         for (int i = q; i <= K; i += p)
80             if (!e[i])
81                 add(i);
82     }
83 while (cnt <= v)
84     w[cnt] = query(cnt), cnt++;
85
86 vector<int> primes;
87 primes.push_back(1);
88 for (int i = 2; i <= v; ++i)
89     if (!e[i])
90         primes.push_back(i);
91 l[1] += i64(w[v] + w[n_4] - 1) * (w[v] - w[n_4]) / 2;
92 for (int i = w[n_4] + 1; i <= w[B]; ++i)
93     l[1] -= l[primes[i]];
94 for (int i = w[B] + 1; i <= w[v]; ++i)
95     l[1] -= query(N / primes[i]);
96 for (int i = w[n_4] + 1; i <= w[v]; ++i) {
97     int q = primes[i];
98     i64 M = N / q;
99     int e = w[M / q];
100     if (e <= i)
101         break;
102     l[1] += e - i;
103     i64 t = 0;
104     int m = w[sqrt(M + 0.5)];
105     for (int k = i + 1; k <= m; ++k)
106         t += w[div(M, primes[k])];
107     l[1] += 2 * t - (i + m) * (m - i);
108 }
109 return l[1];
}

```

```

33     return 1;
34 }
35 return 0;
36 }
37 bool miller_rabin(const uLL x) {
38     if (x == 1)
39         return 0;
40     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. 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}; /*2^64*/
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)

```

## 2.12. Formula

### 2.12.1. Dirichlet Convolution

$$\varepsilon = \mu * 1$$

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

### 2.12.2. 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

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

### 2.12.3. Pick Theorem

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

### 2.12.4. Fermat's Little Theorem

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



### 2.12.5. Wilson's Theorem

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

### 2.12.6. Legendre Theorem

$v(n) := \text{power of } p \text{ in } n$

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

$s(n) := \text{sum of all digits of } n \text{ in base } p$

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

### 2.12.7. Kummer Theorem

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

### 2.12.8. ext-Kummer Theorem

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

### 2.12.9. 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.

### 2.12.10. Properties of nCr with mod

If any  $i$  in base  $p$  satisfies  $n_i < m_i$ , then  $\binom{n}{m}_p \% p = 0$ . Therefore  $\binom{n}{m} = \prod_{i=0}^{\max(\log_p(a), \log_p(b))} \binom{n_i}{m_i} \% p$  so  $\binom{n}{m} \% p = 0$ . 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.

### 2.12.11. 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.

### 2.12.12. Catalan Number

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

### 2.12.13. modinv table

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

### 2.13. 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)

```

```

41         for (int j = l; j < m; ++j)
42             scanf("%lld", &v[i][j]);
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;

```

```

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     v = ans.v;
228 }
229 return *this;
230 }
231 Matrix operator*=(T x) {
232     int n = v.size(), m = v[0].size();
233     for (int i = 0; i < n; ++i)
234         for (int j = 0; j < m; ++j) {
235             v[i][j] *= x;
236             if (MOD) {
237                 if (v[i][j] < 0)
238                     v[i][j] = (v[i][j] % MOD + MOD) % MOD;
239                 if (v[i][j] >= MOD)
240                     v[i][j] %= MOD;
241             }
242         }
243     return *this;
244 }
245 Matrix operator/(T x) {
246     Matrix ans(*this);
247     int n = v.size(), m = v[0].size();
248     for (int i = 0; i < n; ++i)
249         for (int j = 0; j < m; ++j) {
250             if (MOD) {
251                 ans.v[i][j] *= modeq(x, (T)1, (T)MOD);
252                 if (ans.v[i][j] < 0)
253                     ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
254                 if (ans.v[i][j] >= MOD)
255                     ans.v[i][j] %= MOD;
256             } else
257                 ans.v[i][j] /= x;
258         }
259     return ans;
260 }
261 Matrix operator/=(T x) {
262     int n = v.size(), m = v[0].size();
263     for (int i = 0; i < n; ++i)
264         for (int j = 0; j < m; ++j) {
265             if (MOD) {
266                 v[i][j] *= modeq(x, (T)1, (T)MOD);
267                 if (v[i][j] < 0)
268                     v[i][j] = (v[i][j] % MOD + MOD) % MOD;
269                 if (v[i][j] >= MOD)
270                     v[i][j] %= MOD;
271             } else
272                 v[i][j] /= x;
273         }
274     return *this;
275 }
276 Matrix operator%=(T p) {
277     int n = v.size(), m = v[0].size();
278     for (int i = 0; i < n; ++i)
279         for (int j = 0; j < m; ++j)
280             if (v[i][j] >= p)
281                 v[i][j] %= p;
282     return *this;
283 }
284 void gaussian() {
285     int curi = 0;
286     int n = v.size();
287     int m = v[0].size();
288     for (int j = 0; j < m; j++) {
289         int i;
290         for (i = curi; i < n; i++) {
291             if (MOD) {
292                 v[i][j] %= MOD;
293             }
294             if (v[i][j]) {
295                 break;
296             }
297         }
298         if (i >= n) {
299             continue;
300         }
301         if (v[i][j] == 0)
302             continue;
303         for (int k = 0; k < m; k++) {
304             swap(v[i][k], v[curi][k]);
305         }
306         for (int k = m - 1; k >= j; k--) {
307             if (MOD) {
308                 v[curi][k] *= modeq(v[curi][j], (T)1, (T)MOD);
309                 v[curi][k] = (v[curi][k] % MOD + MOD) % MOD;
310             } else
311                 v[curi][k] /= v[curi][j];
312         }
313         for (int i = 0; i < n; ++i) {
314             if (i != curi) {
315                 for (int k = m - 1; k >= j; k--) {
316                     v[i][k] -= v[curi][k] * v[i][j];
317                     if (MOD) {
318                         v[i][k] = (v[i][k] % MOD + MOD) % MOD;
319                     }
320                 }
321             }
322         }
323     }
324 }

```



```

    }
    curi++;
}
};

```

### 2.14. Miller Rabin

```

1 #define uLL __uint128_t
2 template <class T, class POW> void fastpow(T x, POW n, POW p, T&ans) {
3     for (; n; n >>= 1) {
4         if (n & 1) {
5             ans *= x;
6             ans %= p;
7         }
8         x *= x;
9         x %= p;
10    }
11 }
12 /* 輸入 x,n,p,ans 會將 ans 修改為 x^n%p
13 對整數/矩陣/不要求精度的浮點 皆有效
14 模板第一個型別是 x,ans 第二個是 n,p(應該放 LL 或 __uint128_t)*/
15 uLL pri[7] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
16 // int p[3]={2,7,61};/*2^32*/
17 bool check(const uLL x, const uLL p) {
18     uLL d = x - 1, ans = 1;
19     fastpow(p, d, x, ans);
20     if (ans != 1)
21         return 1;
22     for (; !(d & 1);) {
23         d >>= 1;
24         ans = 1;
25         fastpow(p, d, x, ans);
26         if (ans == x - 1)
27             return 0;
28         else if (ans != 1)
29             return 1;
30     }
31     return 0;
32 }
33 bool miller_rabin(const uLL x) {
34     if (x == 1)
35         return 0;
36     for (auto e : pri) {
37         if (e >= x)
38             return 1;
39         if (check(x, e))
40             return 0;
41     }
42     return 1;
43 }

```

### 2.15. Xor Basis

```

1 #pragma GCC optimize("Ofast,fast-math,unroll-loops,no-stack-protector")
2 using namespace std;
3 #define ll long long
4 #define V vector
5 #define pb push_back
6 #define all(x) x.begin(), x.end()
7 V<ll> v;
8 ll f(ll k, ll now = 0, ll p = v.size() - 1, ll ans = 0) {
9     if (k >= 1 << p) {
10         k -= 1 << p;
11         ans = max(ans, ans ^ v[now]);
12     } else
13         ans = min(ans, ans ^ v[now]);
14     if (!p)
15         return ans;
16     return f(k, now + 1, p - 1, ans);
17 }
18 int main() {
19     ios::sync_with_stdio(0);
20     cin.tie(0);
21     cout.tie(0);
22     ll n, k;
23     cin >> n >> k;
24     for (ll x, i = 0; i < n; ++i) {
25         cin >> x;
26         for (ll &e : v)
27             x = min(x, x ^ e);
28         if (x)
29             v.pb(x);
30     }
31     sort(all(v), greater<ll>());
32     ll t = n - v.size(), a = k >> t, b = k & ((1 << min(t, 20LL)) - 1), i = 0;
33     for (; a--; ++i)
34         for (ll j = 1 << t, p = f(i); j--;)
35             cout << p << " ";
36     for (i = f(i); b--;)
37         cout << i << " ";
38 }

```

## 3. String

### 3.1. KMP

```

1 string s, t;
2 int pmt[1000005];
3
4 void init() {
5     for (int i = 1, j = 0; i < t.size(); i++) {
6         while (j && t[j] != t[i]) {
7             j = pmt[j - 1];
8         }
9         if (t[j] == t[i])
10             j++;
11         pmt[i] = j;
12     }
13 }
14
15 int kmp(string s) {
16     int ret = 0;
17     for (int i = 0, j = 0; i < s.size(); i++) {
18         while (j && s[i] != t[j]) {
19             j = pmt[j - 1];
20         }
21         if (s[i] == t[j]) {
22             j++;
23         }
24         if (j == t.size()) {
25             ret++;
26             j = pmt[j - 1];
27         }
28     }
29     return ret;
30 }

```

### 3.2. Longest Palindrome

```

1 #define int long long
2 using namespace std;
3
4 string s;
5 string t;
6 int n;
7 int d[2000005];
8 int ans = 0;
9
10 signed main() {
11     cin >> t;
12     n = t.size();
13     for (int i = 0; i < 2 * n + 1; i++) {
14         if (i & 1) {
15             s += '0';
16         } else {
17             s += t[i / 2];
18         }
19     }
20     n = s.size();
21     d[0] = 1;
22     for (int i = 0, l = 0, r = 0; i < n; i++) {
23         if (i > r) {
24             d[i] = 1;
25             bool a = i + d[i] < n;
26             bool b = i - d[i] >= 0;
27             bool c = (s[i + d[i]] == s[i - d[i]]);
28             while (a && b && c) {
29                 d[i]++;
30                 a = i + d[i] < n;
31                 b = i - d[i] >= 0;
32                 c = (s[i + d[i]] == s[i - d[i]]);
33             }
34             l = i - d[i] + 1;
35             r = i + d[i] - 1;
36         } else {
37             int j = l + r - i;
38             if (j - d[j] + 1 > l) {
39                 d[i] = d[j];
40             } else {
41                 d[i] = r - i + 1;
42                 a = i + d[i] < n;
43                 b = i - d[i] >= 0;
44                 c = (s[i + d[i]] == s[i - d[i]]);
45                 while (a && b && c) {
46                     d[i]++;
47                     a = i + d[i] < n;
48                     b = i - d[i] >= 0;
49                     c = (s[i + d[i]] == s[i - d[i]]);
50                 }
51                 l = i - d[i] + 1;
52                 r = i + d[i] - 1;
53             }
54         }
55         // cout << d[i] << " ";
56     }
57 }

```

```

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

```

### 3.4. Booth

```

1  #define V vector
3  string booth(string s) {
4      s += s;
5      int n = s.size(), k = 0;
6      V<int> f(n, -1);
7      for (int i = 1; i < n; ++i) {
8          int j = f[i - k - 1];
9          for (; j >= 0 && s[j + k + 1] != s[i]; j = f[j])
10             if (s[i] < s[j + k + 1])
11                 k = i - j - 1;
12             if (s[i] != s[j + k + 1]) {
13                 if (s[i] < s[k])
14                     k = i;
15                 f[i - k] = -1;
16             } else
17                 f[i - k] = j + 1;
18         }
19     }
20     return s.substr(k, s.size() >> 1);
21 }
22 //給出循環排列後最小字典序的解

```

## 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) {
34     if (vis2[x])
35         return;
36     vis2[x] = 1;
37     for (int i : v[x]) {
38         dfs(i);
39     }
40 }
41 }
42 void dfs2(int x) {
43     if (cycle[x] || d[x])
44         return;
45     dfs2(a[x]);
46     d[x] = d[a[x]] + 1;
47     r[x] = r[a[x]];
48     cycle[x] = cycle[a[x]];
49 }
50 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++) {
72         cout << d[i] + len[cycle[i]] << " ";
73     }
74 }

```

### 4.2. Dijkstra

```

1  vector<pair<int, int>> v[100005], v2[100005];
2  vector<edge> es;
3  int dis1[100005];
4  int dis2[100005];
5  bitset<100005> vis1, vis2;
7  void dijkstra(int x, int *dis, vector<pair<int, int>> *v, bitset<100005> vis,
8  priority_queue<pair<int, int>, vector<pair<int, int>>,
9  greater<pair<int, int>>> pq) {
10     pq;
11     memset(dis, 0x3f, sizeof(dis));
12     vis.reset();
13     dis[x] = 0;
14     pq.push({0, x});
15     while (!pq.empty()) {
16         pair<int, int> now = pq.top();
17         pq.pop();
18         if (vis[now.second])
19             continue;
20         vis[now.second] = 1;
21         for (auto [i, w] : v[now.second]) {
22             if (vis[i])
23                 continue;
24             if (dis[now.second] + w < dis[i]) {
25                 dis[i] = dis[now.second] + w;
26                 pq.push({dis[i], i});
27             }
28         }
29     }
30 }

```

## 4.3. MaximumFlow

```

1  #define int long long
3  using namespace std;

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

11 bool bfs() {
13     memset(head, 0, sizeof(head));
    memset(lv, 0, sizeof(lv));
15     queue<int> q;
    q.push(1);
17     while (!q.empty()) {
        int now = q.front();
19         q.pop();
        if (now == n)
21             continue;
        for (int i : v[now]) {
23             if (i != 1 && c[now][i] && !lv[i]) {
                lv[i] = lv[now] + 1;
25                 q.push(i);
            }
27         }
29     }
    return lv[n];
31 }

33 int dfs(int x, int flow) {
    int ret = 0;
35     if (x == n)
        return flow;
    for (int i = head[x]; i < v[x].size(); i++) {
37         int y = v[x][i];
        head[x] = y;
39         if (c[x][y] && lv[y] == lv[x] + 1) {
            int d = dfs(y, min(flow, c[x][y]));
41             flow -= d;
            c[x][y] -= d;
43             c[y][x] += d;
            ret += d;
45         }
    }
47     return ret;
49 }

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

```

## 4.4. Dinic

```

1  using namespace std;
2  #define ll long long
3  const ll inf = 8e18;
4  #define N 505
5  #define pb push_back
7  struct pp {
    int from, to;
9     ll flow;
};

11 int t, lvl[N], p[N];
vector<int> g[N];
13 vector<pp> edge;
int bfs(int s) {
15     queue<int> q;
    for (q.push(s), lvl[s] = 1; !q.empty(); q.pop()) {
17         int u = q.front();
        for (int e : g[u]) {
19             int v = edge[e].to;
            if (lvl[v] || !edge[e].flow)
21                 continue;

```

```

        lvl[v] = lvl[u] + 1;
        q.push(v);
    }
25     }
    return lvl[t];
27 }

29 ll dfs(int u, ll f = inf) {
    if (u == t || !f)
31         return f;
    ll ans = 0;
    for (int &i = p[u]; i < g[u].size(); ++i) {
33         pp &e = edge[g[u][i]], &b = edge[g[u][i] ^ 1];
        if (lvl[e.to] == lvl[u] + 1) {
35             ll c = dfs(e.to, min(e.flow, f));
            e.flow -= c;
37             b.flow += c;
            f -= c;
39             ans += c;
        }
41     }
    return ans;
43 }

45 ll dinic(int s) {
    ll ans = 0;
    for (; bfs(s); memset(lvl, 0, sizeof lvl))
47         for (ll k; k = (memset(p, 0, sizeof(p)), dfs(s));)
            ans += k;
49     return ans;
51 }

53 int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
55     int n, m, cnt = 0;
    for (cin >> n >> m; m--;) {
57         int u, v;
        ll f;
59         cin >> u >> v >> f;
        g[u].pb(cnt++);
61         g[v].pb(cnt++);
        edge.pb({u, v, f});
63         edge.pb({v, u, 0});
    }
65     t = n;
    cout << dinic(1);
67 }

```

## 4.5. SCC

```

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

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

```

## 4.6. VBCC

```

1  using namespace std;
2  #define pb push_back
3  #define pii pair<int, int>
4  #define N 100005
5  vector<int> adj[N], bcc[N];
7  stack<int> st;
int dfn[N], low[N], tag, bc, root;

```

```

9 bitset<N> ap;
void dfs(int now, int par = -1) {
11     st.push(now);
    low[now] = dfn[now] = ++tag;
13     int f = 0;
    for (int e : adj[now] | views::reverse) {
15         if (e == par)
            continue;
17         if (!dfn[e]) {
            dfs(e, now), low[now] = min(low[now], low[e]);
19             if (low[e] >= dfn[now]) {
                 if (++f > 1 || now != root)
                     ap[now] = 1;
                     ++bc;
23                 for (; st.top() != now; st.pop())
                     bcc[bc].pb(st.top());
                     bcc[bc].pb(now);
25             }
        } else
            low[now] = min(low[now], dfn[e]);
27     }
29 }
31 int main() {
    int n, m, u, v;
33     cin >> n >> m;
    vector<pii> g(m);
35     for (auto &[u, v] : g)
        cin >> u >> v, adj[u].pb(v), adj[v].pb(u);
37     for (root = 1; root <= n; ++root)
        if (!dfn[u])
            dfs(root);
39     int ans = 0;
41     for (int i : views::iota(1) | views::take(n))
        if (ap[i])
            ++ans;
43     cout << ans << "\n";
45     for (int i : views::iota(1) | views::take(n))
        if (ap[i])
            cout << i << " ";
47 }

```

#### 4.7. 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;
21     return x + m;
23 }
25 void dfs(int x) {
    d[x] = low[x] = ++cnt;
27     s.push(x);
    for (int i : v[x]) {
29         if (scc[i])
            continue;
31         if (d[i]) {
            low[x] = min(low[x], d[i]);
33         } else {
            dfs(i);
35             low[x] = min(low[x], low[i]);
        }
37     }
    if (d[x] == low[x]) {
39         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);
53     cout.tie(0);
    cin >> n >> m;
55     while (n--) {
        char a, b;
57         int x, y;
        cin >> a >> x >> b >> y;
59         if (a == '-')
            x = no(x);
61         if (b == '-')
            y = no(y);
63         v[no(x)].push_back(y);
        v[no(y)].push_back(x);
65     }
    for (int i = 1; i <= 2 * m; i++) {
67         if (!d[i]) {
            dfs(i);
69         }
71     }
    for (int i = 1; i <= m; i++) {
73         if (scc[i] ^ scc[i + m]) {
            op[scc[i]] = scc[i + m];
75             op[scc[i + m]] = scc[i];
        } else {
77             cout << "IMPOSSIBLE";
            exit(0);
79         }
    }
    for (int i = 1; i <= 2 * m; i++) {
81         for (int j : v[i]) {
            if (scc[i] ^ scc[j]) {
83             v2[scc[j]].push_back(scc[i]);
            ind[scc[i]]++;
85         }
        }
87     }
    for (int i = 1; i <= now; i++) {
89         if (!ind[i]) {
            q.push(i);
91         }
93     }
    while (!q.empty()) {
95         int k = q.front();
        q.pop();
97         if (!ans[k]) {
            ans[k] = 1;
99             ans[op[k]] = 2;
        }
        for (int i : v2[k]) {
101             ind[i]--;
            if (!ind[i]) {
103                 q.push(i);
            }
105         }
    }
    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];
9 int f[200005];
int dp[200005];
11
void update(line ins, int l = 1, int r = 1e6, int index = 1) {
13     if (l == r) {
        if (ins.y(l) < tree[index].y(l)) {
15             tree[index] = ins;
        }
        return;
17     }
    int mid = (l + r) >> 1;
19     if (tree[index].a < ins.a)
        swap(tree[index], ins);
21     if (tree[index].y(mid) > ins.y(mid)) {
        swap(tree[index], ins);
23     }

```

```

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

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

```

## 5.2. CHO

```

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

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})) {
            dq.pop_back();
23        }
        dq.push_back({a, b});
25    }

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

```

## 5.3. SOSDP

```

1 for (int i = 0; i < 20; ++i)
    for (int j = i; j < N; ++j)
3         if (j >> i & 1)
            dp[j] += dp[j ^ (1 << i)]; // subset
5 for (int i = 0; i < 20; ++i)
    for (int j = 0; j < N; ++j)
7         if (!(j >> i & 1))
            dp2[j] += dp2[j | (1 << i)]; // superset

```

## 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;
}

```

```

19     return -1;
}

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

```

### 6.2. Inside

```

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

```

### 6.3. Minimum Euclidean Distance

```

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

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

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

15 signed main() {
    ios::sync_with_stdio(0);
17     cin.tie(0);
    cout.tie(0);
19     cin >> n;
    for (int i = 0; i < n; i++) {
21         int x, y;
        cin >> x >> y;
23         x += 1000000000;
        v.push_back({x, y});
25     }

    sort(v.begin(), v.end());
27     int l = 0;
    for (int i = 0; i < n; i++) {
29         int d = ceil(sqrt(dd));
        while (l < i && v[l].first - v[i].first > d) {
31             s.erase({v[l].second, v[l].first});
            l++;
33         }

        auto x = s.lower_bound({v[i].second - d, 0});
        auto y = s.upper_bound({v[i].second + d, 0});
35         for (auto it = x; it != y; it++) {
            dd = min(dd, dis(*it, v[i]));
37         }
        s.insert({v[i].second, v[i].first});
39     }
    cout << dd;
41 }
43 }

```

### 6.4. Convex Hull

```

1 #define int long long
3 #define fastio
    ios_base::sync_with_stdio(0);
5     cin.tie(0);
    cout.tie(0);

7 using namespace std;

9 template <typename T> pair<T, T> operator-(pair<T, T> a, pair<T, T> b) {
    return {a.first - b.first, a.second - b.second};
}

```

```

11 return make_pair(a.first - b.first, a.second - b.second);
13 }
15 template <typename T> T cross(pair<T, T> a, pair<T, T> b) {
16     return a.first * b.second - a.second * b.first;
17 }
19 template <typename T> vector<pair<T, T>> getCH(vector<pair<T, T>> v) {
20     int n = v.size();
21     sort(v.begin(), v.end());
22     vector<pair<T, T>> hull;
23     for (int i = 0; i < 2; i++) {
24         int t = hull.size();
25         for (auto x : v) {
26             while (hull.size() - t >= 2 &&
27                    cross(hull[hull.size() - 1] - hull[hull.size() - 2],
28                          x - hull[hull.size() - 2]) <= 0)
29                 hull.pop_back();
30             hull.push_back(x);
31         }
32         hull.pop_back();
33         reverse(v.begin(), v.end());
34     }
35     return hull;
36 }
37
38 if (dot(p1, p2, p3) < 0 || dot(p2, p1, p3) < 0)
39     return min(len(p3 - p1), len(p3 - p2));
40 return abs(cross(p1, p2, p3)) / len(p2 - p1);
41 }
42 ll area2(
43     vector<pll> &
44     v) { //傳入一個多邊形照順序的點集，起點要出現兩次，回傳兩倍面積，注意
45     // ll
46     int n = v.size() - 1;
47     ll ans = 0;
48     for (int i = 0; i < n; ++i)
49         ans += cross(v[i], v[i + 1]);
50     return abs(ans);
51 }
52 int in_polygon(vector<pll> &v,
53                pll p) { //傳入多邊形，起點要出現兩次，回傳 {-1:in, 0:on, 1:out}
54     int n = v.size() - 1, ans = 1;
55     for (int i = 0; i < n; ++i)
56         if (btw(v[i], v[i + 1], p))
57             return 0;
58     for (int i = 0; i < n; ++i)
59         if (banana(v[i], v[i + 1], p, {(ll)2e9 + 7, p.S + 1LL}))
60             ans *= -1;
61     //對於任意 p 到 {W, p.S+1} 的向量中不會有整數點存在，其中需要滿足 {W, p.S}
62     //必須很遠，保證在多邊形外
63     return ans;
64 }
65 void solve() {
66     int n;
67     cin >> n;
68     vector<pll> v(n);
69     for (pll &e : v)
70         cin >> e;
71     v.pb(v[0]);
72     ll ans = area2(v) + 2, ans2 = 0;
73     for (int i = 0; i < n; ++i) {
74         if (v[i].F == v[i + 1].F)
75             ans2 += abs(v[i].S - v[i + 1].S);
76         else if (v[i].S == v[i + 1].S)
77             ans2 += abs(v[i].F - v[i + 1].F);
78         else
79             ans2 += gcd(abs(v[i].F - v[i + 1].F), abs(v[i].S - v[i + 1].S));
80     }
81     cout << (ans - ans2) / 2 << " " << ans2;
82 }
83 int main() {
84     int t = 1;
85     // cin>>t;
86     for (; t--;) {
87         solve();
88     }
89 }

```

## 6.5. 164253 Version

```

1 using namespace std;
2 #define ll long long
3 #define pb push_back
4 #define pll pair<int, int>
5 #define pdd pair<double, double>
6 #define pll pair<ll, ll>
7 #define F first
8 #define S second
9 #define eps 1e-6
10 int sign(double x) { return fabs(x) < eps ? 0 : x > 0 ? 1 : -1; }
11 int sign(ll x) { return !x ? 0 : x > 0 ? 1 : -1; }
12 template <typename T1, typename T2>
13 istream &operator>>(istream &s, pair<T1, T2> &p) {
14     auto &a, &b = p;
15     s >> a >> b;
16     return s;
17 }
18 template <typename T1, typename T2>
19 ostream &operator<<(ostream &s, const pair<T1, T2> p) {
20     auto &a, &b = p;
21     s << a << " " << b;
22     return s;
23 }
24 pll operator+(const pll a, const pll b) { return {a.F + b.F, a.S + b.S}; }
25 pll operator-(const pll a, const pll b) { return {a.F - b.F, a.S - b.S}; }
26 pll operator*(const pll a, const pll b) { return {-a.F, -a.S}; }
27 pll operator*(const pll a, const pll b) {
28     return {(ll)a.F * b.F, (ll)a.S * b.S};
29 }
30 pdd operator/(const pll a, const double x) { return {a.F / x, a.S / x}; }
31 pdd operator*(const pll a, const double x) { return {a.F * x, a.S * x}; }
32 pdd operator*(const double x, const pll a) { return {a.F * x, a.S * x}; }
33 //沒有標示幾個 vector 的都是對三個點做事，以第一個點為參考點
34 ll len2(pll p) { return (ll)p.F * p.F + (ll)p.S * p.S; } // 1 vector
35 double len(pll p) { return sqrt((double)len2(p)); }
36 ll cross(pll a, pll b) { return (ll)a.F * b.S - (ll)a.S * b.F; }
37 ll cross(pll p1, pll p2, pll p3) {
38     return cross(p2 - p1, p3 - p1);
39 } // (b-a) cross (c-a)
40 ll dot(pll a, pll b, pll c) {
41     return (ll)(b.F - a.F) * (c.F - a.F) + (ll)(b.S - a.S) * (c.S - a.S);
42 } // (b-a) dot (c-a)
43 ll ori(pll p1, pll p2, pll p3) {
44     return sign(cross(p1, p2, p3));
45 } // normalize to {-1,0,1} (b-a) cross (c-a)
46 bool btw(pll p1, pll p2, pll p3) {
47     return ori(p3, p1, p2) == 0 && dot(p3, p1, p2) <= 0;
48 } // p3 between p1,p2
49 bool banana(pll p1, pll p2, pll p3, pll p4) { //問兩線段是否香蕉
50     if (btw(p1, p2, p3) || btw(p1, p2, p4) || btw(p3, p4, p1) || btw(p3, p4, p2))
51         return true;
52     return ori(p1, p2, p3) * ori(p1, p2, p4) < 0 &&
53            ori(p3, p4, p1) * ori(p3, p4, p2) < 0;
54 }
55 pdd banana_point(
56     pll p1, pll p2, pll p3,
57     pll p4) { //分點，算的是無限延伸直線的交點，平行的時候 undefined
58     return cross(p2 - p1, p4 - p1) / (double)cross(p2 - p1, p4 - p3) * p3 -
59            cross(p2 - p1, p3 - p1) / (double)cross(p2 - p1, p4 - p3) * p4;
60 }
61 pdd proj(pll p1, pll p2, pll p3) {
62     return dot(p1, p2, p3) / (double)len2(p2 - p1) * (p2 - p1);
63 }
64 double min_dis(pll p1, pll p2, pll p3) { // min distance of p3 to segment p1,p2

```

## 7. Tree

### 7.1. Heavy Light Decomposition (modify and query)

```

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

```



```

35     if (R <= mid) {
36         return query(L, R, l, mid, index << 1);
37     }
38     if (L > mid) {
39         return query(L, R, mid + 1, r, index << 1 | 1);
40     }
41     return max(query(L, mid, l, mid, index << 1),
42               query(mid + 1, R, mid + 1, r, index << 1 | 1));
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 void dfs(int x, int pre) {
58     si[x] = 1;
59     for (int i : v[x]) {
60         if (i == pre)
61             continue;
62         p[i] = x;
63         d[i] = d[x] + 1;
64         dfs(i, x);
65         si[x] += si[i];
66     }
67 }
68 void dfs2(int x, int pre, int t) {
69     tp[x] = t;
70     st[x] = ++cnt;
71     int ma = 0;
72     for (int i : v[x]) {
73         if (i == pre)
74             continue;
75         if (si[i] > si[ma]) {
76             ma = i;
77         }
78     }
79     if (!ma)
80         return;
81     dfs2(ma, x, t);
82     for (int i : v[x]) {
83         if (i == pre || i == ma)
84             continue;
85         dfs2(i, x, i);
86     }
87 }
88 int f(int x, int y) {
89     int ret = 0;
90     while (tp[x] ^ tp[y]) {
91         if (d[tp[x]] < d[tp[y]]) {
92             swap(x, y);
93         }
94         ret = max(ret, query(st[tp[x]], st[x]));
95         x = p[tp[x]];
96     }
97     if (d[x] > d[y])
98         swap(x, y);
99     ret = max(ret, query(st[x], st[y]));
100     return ret;
101 }
102 signed main() {
103     ios::sync_with_stdio(0);
104     cin.tie(0);
105     cout.tie(0);
106     cin >> n >> q;
107     for (int i = 1; i <= n; i++) {
108         cin >> a[i];
109     }
110     for (int i = 1; i < n; i++) {
111         int x, y;
112         cin >> x >> y;
113         v[x].push_back(y);
114         v[y].push_back(x);
115     }
116     dfs(1, 0);
117     dfs2(1, 0, 1);
118     for (int i = 1; i <= n; i++) {
119         b[st[i]] = a[i];
120     }
121     build();

```

```

127     while (q--) {
128         int mode, x, y;
129         cin >> mode >> x >> y;
130         if (mode == 1) {
131             modify(st[x], y);
132         } else {
133             cout << f(x, y) << " ";
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         }
44         return a[x][0];
45     }
46
47     signed main() {
48         ios::sync_with_stdio(0);
49         cin.tie(0);
50         cout.tie(0);
51         cin >> n >> q;
52         for (int i = 1; i < n; i++) {
53             int x, y;
54             cin >> x >> y;
55             v[x].push_back(y);
56             v[y].push_back(x);
57         }
58         dfs(1, 0);
59         init();
60         while (q--) {
61             int x, y;
62             cin >> x >> y;
63             int k = lca(x, y);
64             cout << (d[x] + d[y] - 2 * d[k]) << "\n";
65         }
66     }

```

## 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) {

```

```

7  double ret = 0;
8  double k = 1;
9  for (int i = 0; i <= n; i++) {
10     ret += k * a[i];
11     k *= x;
12 }
13 return -ret;
14 }
15
16 template <class T> T bi_search(T l, T r, T end) {
17     if (!check(r - end))
18         return r - end;
19     for (; r - l > end; ) {
20         T mid = (l + r) / 2;
21         if (check(mid))
22             r = mid;
23         else
24             l = mid;
25     }
26     return l;
27 }
28 /*check gives 000000001111 find the last 0*/
29
30 template <class T> T tri_search(T l, T r, T end) {
31     T midl, midr;
32     for (;;) {
33         midl = (l + r) / 2;
34         midr = (midl + r) / 2;
35         if (midr - midl < end)
36             break;
37         if (get(midr) > get(midl))
38             r = midr;
39         else
40             l = midl;
41     }
42     for (; r - l > end; ) {
43         midl = (l + r) / 2;
44         if (get(r) > get(l))
45             r = midl;
46         else
47             l = midl;
48     }
49     return l;
50 }
51 /*get gives the value, find the minimum*/
52
53 int main() {
54     cin >> n >> x >> y;
55     for (int i = n; i >= 0; i--) {
56         cin >> a[i];
57     }
58     cout << fixed << setprecision(7) << tri_search<double>(x, y,
59 }

```

## 8.2. Big Number

```

1 //洛谷 P1005
2
3 using namespace std;
4 #define N 85
5 #define LL long long
6 #define pii pair<int, int>
7 #define F first
8 #define S second
9 struct num {
10     const static LL base = 1000000000LL; // base 1e9
11     LL p[505], len;
12     num() {
13         memset(p, 0, sizeof(p));
14         len = 0;
15     }
16     num(LL x) {
17         memset(p, 0, sizeof(p));
18         len = 0;
19         for (p[len++] = x; p[len - 1] >= base; ++len)
20             p[len] = p[len - 1] / base, p[len - 1] %= base;
21     }
22     num operator=(LL x) {
23         memset(p, 0, sizeof(p));
24         len = 0;
25         for (p[len++] = x; p[len - 1] >= base; ++len)
26             p[len] = p[len - 1] / base, p[len - 1] %= base;
27         return *this;
28     }
29     num max(const num &b) {
30         if (len != b.len)
31             return len > b.len ? *this : b;
32         for (int i = len; i--;)
33             if (p[i] != b.p[i])
34                 return p[i] > b.p[i] ? *this : b;
35         return *this;
36     }
37     num operator+(const num &b) {

```

```

38     num c;
39     LL x = 0;
40     for (LL &i = c.len; i < len || i < b.len; ++i) {
41         c.p[i] = p[i] + b.p[i] + x;
42         x = c.p[i] / base;
43         c.p[i] %= base;
44     }
45     if (x)
46         c.p[c.len++] = x;
47     return c;
48 }
49 num operator*(LL b) {
50     num c;
51     c.len = len;
52     LL x = 0;
53     for (LL i = 0; i < len; ++i) {
54         c.p[i] = p[i] * b + x;
55         x = c.p[i] / base;
56         c.p[i] %= base;
57     }
58     for (; x; x /= base)
59         c.p[c.len++] = x % base;
60     return c;
61 }
62 } dp[N][N], ans;
63 ostream &operator<<(ostream &s, num a) {
64     if (!a.len)
65         return s << "0";
66     s << a.p[a.len - 1];
67     for (int i = a.len - 1; i--;) {
68         if (!a.p[i])
69             s << "000000000";
70         else {
71             for (int k = 10; k * a.p[i] < (LL)1e9; k *= 10)
72                 s << "0";
73             s << a.p[i];
74         }
75     }
76     return s;
77 }
78 LL a[N];
79 int main() {
80     ios::sync_with_stdio(0);
81     cin.tie(0);
82     cout.tie(0);
83     int n, m, i, j;
84     for (cin >> n >> m; n--;) {
85         for (i = 0; i < m; ++i)
86             cin >> a[i];
87         for (i = 0; i < m; ++i)
88             for (j = 0; j < m; ++j)
89                 dp[i][j] = 0;
90         for (i = 0; i < m; ++i)
91             dp[i][i] = a[i] << 1;
92         for (j = 1; j < m; ++j)
93             for (i = 0; i + j < m; ++i)
94                 dp[i][i + j] =
95                     (dp[i][i + j - 1] + a[i + j]).max(dp[i + 1][i + j]) + a[i + j];
96         ans = ans + dp[0][m - 1];
97     }
98     cout << ans;
99 }

```

## 8.3. 對拍

```

1 script
2 ```bash
3 set -e
4 g++ ac.cpp -o ac
5 g++ wa.cpp -o wa
6 for ((i=0;;i++))
7 do
8     echo "$i"
9     python3 gen.py > input
10    ./ac < input > ac.out
11    ./wa < input > wa.out
12    diff ac.out wa.out || break
13 done
14 # factor n 可以質因數分解
15 ```
16
17 python random
18 ```python
19 from random import *
20 n = randint(1, 100)
21 ch = chr(ord('a') + randint(0, 25))
22 choiceSet = sample(s, 4)
23 choiceSet = sample(range(1, n+1), 4)
24 shuffle(arr)
25 ```
26
27 python tree
28 ```python

```

# 隨機產生 1~100 的整數  
# 隨機產生 'a'~'z' 其中一個  
# 從集合 s 選出 4 個不同的元  
# 從整數 1~n 選出 4 個不同的  
# 把序列 arr 順序打亂

```

29 from random import *
   n = randint(3, 6)
31 print(n)
   for i in range(2, n+1):
33     print(randint(1, i-1), i)
35
37 簡單連通圖
   ```python
39 from random import *
   n = randint(5, 10)
41 m = randint(n-1, n+3)
43
45 print(n, m)
47 edge = list()
49
51 #construct tree
   for i in range(2, n+1):
49     x = randint(1, i-1)
       y = i
       edge.append([min(x, y), max(x, y)])
       print(x, y)
53
55 #add extra edge
   for i in range(m-(n-1)):
       x = randint(1, n)
       y = randint(1, n)
       while x == y or [min(x, y), max(x, y)] in edge:
59         x = randint(1, n)
           y = randint(1, n)
       print(x, y)
       edge.append([min(x, y), max(x, y)])
63
65 c++ debug template
   ```cpp
67 #ifdef LOCAL // ===== Local =====
   void dbg() { cerr << '\n'; }
69 template<class T, class ...U> void dbg(T a, U ...b) { cerr << a << ' ', dbg(b...); }
   template<class T> void org(T l, T r) { while (l != r) cerr << *l++ << ' '; cerr << '\n'; }
71 #define debug(args...) (dbg("#> (" + string(#args) + ") = (" + args, ")"))
   #define orange(args...) (cerr << "#> [" + string(#args) + ") = ", org(args))
73 #else // ===== OnlineJudge =====
   #pragma GCC optimize("O3,unroll-loops")
75 #pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
   #define debug(...) ((void)0)
77 #define orange(...) ((void)0)
   #endif
79

```