Contents

1. DataStructure

1.1. Treap

```cpp
#define pii pair<int, int>
struct node {
  int tag = 0;
  int sum = 0;
  int prio = rand();
  int lson = 0;
  int rson = 0;
  int si = 0;
  int val = 0;
};
node treap[400005];
int cnt = 0;
int root = 0;

void update(int index) {
  int lson = treap[index].lson;
  int rson = treap[index].rson;
  treap[index].si = treap[lson].si + treap[rson].si + 1;
  treap[index].sum = treap[lson].sum;
  treap[index].sum += treap[rson].sum;
  treap[index].sum += treap[index].val;
}
void push(int index) {
  if (!treap[index].tag)
    return;
  swap(treap[index].lson, treap[index].rson);
  int lson = treap[index].lson;
  int rson = treap[index].rson;
  treap[lson].tag ^= 1;
  treap[rson].tag ^= 1;
  treap[index].tag = 0;
}

pii split(int rk, int index) {
  if (!index)
    return {0, 0};
  push(index);
  int lson = treap[index].lson;
  int rson = treap[index].rson;
  if (rk <= treap[lson].si) {
    pii temp = split(rk, lson);
    treap[index].lson = temp.second;
    update(index);
    return {temp.first, index};
  } else {
    pii temp = split(rk - treap[lson].si - 1, rson);
    treap[index].rson = temp.first;
    update(index);
    return {index, temp.second};
  }
}

int merge(int x, int y) {
  if (!x && !y)
    return 0;
  if (!x && y)
    return y;
  if (x && !y)
    return x;
  push(x);
  push(y);
  if (treap[x].prio < treap[y].prio) {
    treap[x].rson = merge(treap[x].rson, y);
    update(x);
    return x;
  } else {
    treap[y].lson = merge(x, treap[y].lson);
    update(y);
    return y;
  }
}

void insert(int x, int v) {
  pii temp = split(x - 1, root);
  cnt++;
  treap[cnt].val = v;
  update(cnt);
  temp.first = merge(temp.first, cnt);
  root = merge(temp.first, temp.second);
}

int query(int l, int r) {
  pii R = split(r, root);
  pii L = split(l - 1, R.first);
  int ret = treap[L.second].sum;
  R.first = merge(L.first, L.second);
  root = merge(R.first, R.second);
  return ret;
}

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

1.2. Dynamic Segment Tree

```cpp

#define int long long
using namespace std;

int n, q;
struct node {
```

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

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

void push(int index) {
  if (!tree[index].lson) {
    tree[index].lson = ++now;
  }
  if (!tree[index].rson) {
    tree[index].rson = ++now;
  }
  int lson = tree[index].lson;
  int rson = tree[index].rson;
  tree[lson].tag += tree[index].tag;
  tree[rson].tag += tree[index].tag;
  tree[index].data = tree[index].rv();
  tree[index].tag = 0;
}

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

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

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

2. Math

2.1. FFT

```

using namespace std;
inline int read() {
  int ans = 0;
  char c = getchar();
  while (!isdigit(c))
    c = getchar();
  while (isdigit(c)) {
    ans = ans * 10 + c - '0';
    c = getchar();
  }
  return ans;
}
typedef complex<double> comp;
const int MAXN = 1000005;
const comp I(0, 1);
const double PI = acos(-1);
comp A[MAXN * 3], B[MAXN * 3], tmp[MAXN * 3], ans[MAXN * 3];
void fft(comp F[], int N, int sgn = 1) {
  if (N == 1)
    return;
  memcpy(tmp, F, sizeof(comp) * N);
  for (int i = 0; i < N; i++)
    *(i % 2 ? F + i / 2 + N / 2 : F + i / 2) = tmp[i];
  fft(F, N / 2, sgn), fft(F + N / 2, N / 2, sgn);
  comp *G = F, *H = F + N / 2;
  comp cur = 1, step = exp(2 * PI / N * sgn * I);
  for (int k = 0; k < N / 2; k++) {
    tmp[k] = G[k] + cur * H[k];
    tmp[k + N / 2] = G[k] - cur * H[k];
    cur *= step;
  }
  memcpy(F, tmp, sizeof(comp) * N);
}
int main() {
  int n = read(), m = read(), N = 1 << __lg(n + m + 1) + 1;
  for (int i = 0; i <= n; ++i)
    A[i] = read();
  for (int i = 0; i <= m; ++i)
    B[i] = read();
  fft(A, N), fft(B, N);
  for (int i = 0; i < N; ++i)
    ans[i] = A[i] * B[i];
  fft(ans, N, -1);
  for (int i = 0; i <= n + m; ++i)
    printf("%d ", int(ans[i].real() / N + 0.1));
  return 0;
}
```

2.2. NTT

```

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

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

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

```
        F[k + l] = ((g - h) % MOD + MOD) % MOD;
        cur = (ll)cur * step % MOD;
      }
  }
  if (sgn == -1) {
    int invN = qpow(N, MOD - 2);
    for (int i = 0; i < N; ++i)
      F[i] = (ll)F[i] * invN % MOD;
  }
}
```

2.3. Gaussian-Jordan

```

#define int long long
using namespace std;

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

// n <= m
void gaussian(double a[105][105], int n, int m) {
  int curi = 0;
  for (int j = 0; j < m; j++) {
    int i;
    for (i = curi; i < n; i++) {
      if (a[i][j]) {
        break;
      }
    }
    if (a[i][j] == 0)
      continue;
    for (int k = 0; k < m; k++) {
      swap(a[i][k], a[curi][k]);
    }
    for (int k = m - 1; k >= j; k--) {
      a[curi][k] /= a[curi][j];
    }
    for (int i = 0; i < n; ++i) {
      if (i != curi) {
        for (int k = m - 1; k >= j; k--) {
          a[i][k] -= a[curi][k] * a[i][j];
        }
      }
    }
    curi++;
  }
}
```

2.4. Mu

```
vector<int> prime;
bitset<1000005> vis;
int n;
int mu[1000005];

void init() {
  for (int i = 2; i <= n; i++) {
    if (!vis[i]) {
      prime.push_back(i);
      mu[i] = -1;
    }
    for (int p : prime) {
      if (i * p > n)
        break;
      vis[i * p] = 1;
      if (i % p == 0) {
        mu[i * p] = 0;
        break;
      } else {
        mu[i * p] = mu[i] * mu[p];
      }
    }
  }
}
```

2.5. Lucas

```
int fact[100005];
int p;

void init() {
  fact[0] = 1;
  for (int i = 1; i <= p; i++) {
    fact[i] = fact[i - 1] * i % p;
  }
}

int inv(int x, int p) {
  if (x == 1)
    return 1;
  return (p - p / x) * inv(p % x, p) % p;
}

int c(int x, int y, int p) {
  if (x < y)
    return 0;
  int k = fact[x] * inv(fact[y], p) % p;
  return k * inv(fact[x - y], p) % p;
}

int lucas(int x, int y, int p) {
  if (x == 0)
    return 1;
  return lucas(x / p, y / p, p) % p * c(x % p, y % p, p) % p;
}
```

2.6. Inv

```
int exgcd(int a, int b, int &x, int &y) {
  if (b == 0) {
    x = 1;
    y = 0;
    return a;
  }
  int d = exgcd(b, a % b, y, x);
  y -= x * (a / b);
  return d;
}

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

2.7. CRT

```

#define int long long
using namespace std;

int n;
int a[15];
int b[15];
int mul = 1;

void exgcd(int a, int b, int &x, int &y) {
  if (b == 0) {
    x = 1;
    y = 0;
    return;
  }
  exgcd(b, a % b, y, x);
  y -= (a / b) * x;
}

int inv(int a, int p) {
  int x, y;
  exgcd(a, p, x, y);
  return x;
}

int ans = 0;

signed main() {
  cin >> n;
  for (int i = 1; i <= n; i++) {
    cin >> a[i] >> b[i];
    mul *= a[i];
  }
  for (int i = 1; i <= n; i++) {
    ans += inv(mul / a[i], a[i]) * (mul / a[i]) % mul * b[i] % mul;
    ans %= mul;
  }
  ans = (ans + mul) % mul;
  cout << ans;
}
```

2.8. Generator

```

#define int long long
using namespace std;

int t;
int n, d;
bitset<1000005> exist;
bitset<1000005> vis;
vector<int> prime;
```

```cpp
int phi[1000005];

void init() {
  phi[1] = 1;
  for (int i = 2; i <= 1000000; i++) {
    if (!vis[i]) {
      prime.push_back(i);
      phi[i] = i - 1;
    }
    for (int j : prime) {
      if (i * j > 1000000)
        break;
      vis[i * j] = 1;
      if (i % j == 0) {
        phi[i * j] = phi[i] * j;
        break;
      } else {
        phi[i * j] = phi[i] * phi[j];
      }
    }
  }
  exist[2] = exist[4] = 1;
  for (int i : prime) {
    if (i == 2)
      continue;
    for (int j = i; j <= 1000000; j *= i) {
      exist[j] = 1;
      if (j * 2 <= 1000000) {
        exist[j << 1] = 1;
      }
    }
  }
}

vector<int> factors(int x) {
  vector<int> v;
  for (int i = 1; i * i <= x; i++) {
    if (x % i == 0) {
      v.push_back(i);
      if (i * i != x) {
        v.push_back(x / i);
      }
    }
  }
  return v;
}

int f(int x, int y, int mod) {
  int ret = 1;
  while (y) {
    if (y & 1) {
      ret *= x;
      ret %= mod;
    }
    x *= x;
    x %= mod;
    y >>= 1;
  }
  return (ret % mod + mod) % mod;
}

vector<int> findroot(int x) {
  vector<int> ret;
  if (!exist[x])
    return ret;
  int phix = phi[x];
  vector<int> fact = factors(phix);
  int fst;
  for (int i = 1;; i++) {
    if (__gcd(i, x) != 1)
      continue;
    bool ok = 1;
    for (int j : fact) {
      if (j != phix && f(i, j, x) == 1) {
        ok = 0;
        break;
      }
    }
    if (ok) {
      fst = i;
      break;
    }
  }
  int now = fst;
  // cout << fst <<"\n";
  for (int i = 1; i <= phix; i++) {
    if (__gcd(i, phix) == 1) {
      ret.push_back(now);
    }
    now *= fst;
    now %= x;
  }
  return ret;
}

signed main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  init();
  cin >> t;
  while (t--) {
    cin >> n >> d;
    vector<int> v = findroot(n);
    sort(v.begin(), v.end());
    cout << v.size() << "\n";
    for (int i = 0; i < v.size(); i++) {
      if (i % d == d - 1) {
        cout << v[i] << " ";
      }
    }
    cout << "\n";
  }
}
```

2.9. Count Primes

```cpp

using namespace std;
using i64 = long long;
i64 count_pi(i64 N) {
  if (N <= 1)
    return 0;
  int v = sqrt(N + 0.5);
  int n_4 = sqrt(v + 0.5);
  int T = min((int)sqrt(n_4) * 2, n_4);
  int K = pow(N, 0.625) / log(N) * 2;
  K = max(K, v);
  K = min<i64>(K, N);
  int B = N / K;
  B = N / (N / B);
  B = min<i64>(N / (N / B), K);

  vector<i64> l(v + 1);
  vector<int> s(K + 1);
  vector<bool> e(K + 1);
  vector<int> w(K + 1);
  for (int i = 1; i <= v; ++i)
    l[i] = N / i - 1;
  for (int i = 1; i <= v; ++i)
    s[i] = i - 1;

  const auto div = [](i64 n, int d) -> int { return double(n) / d;
  int p;
  for (p = 2; p <= T; ++p)
    if (s[p] != s[p - 1]) {
      i64 M = N / p;
      int t = v / p, t0 = s[p - 1];
      for (int i = 1; i <= t; ++i)
        l[i] -= l[i * p] - t0;
      for (int i = t + 1; i <= v; ++i)
        l[i] -= s[div(M, i)] - t0;
      for (int i = v, j = t; j >= p; --j)
        for (int l = j * p; i >= l; --i)
          s[i] -= s[j] - t0;
      for (int i = p * p; i <= K; i += p)
        e[i] = 1;
    }
  e[1] = 1;
  int cnt = 1;
  vector<int> roughs(B + 1);
  for (int i = 1; i <= B; ++i)
    if (!e[i])
      roughs[cnt++] = i;
  roughs[cnt] = 0x7fffffff;
  for (int i = 1; i <= K; ++i)
    w[i] = e[i] + w[i - 1];
  for (int i = 1; i <= K; ++i)
    s[i] = w[i] - w[i - (i & -i)];

  const auto query = [&](int x) -> int {
    int sum = x;
    while (x)
      sum -= s[x], x ^= x & -x;
    return sum;
  };
  const auto add = [&](int x) -> void {
    e[x] = 1;
    while (x <= K)
      ++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[1] -= l[primes[i]];
  for (int i = w[B] + 1; i <= w[v]; ++i)
    l[1] -= 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

```

using namespace std;
#define LL long long
#define uLL __uint128_t
#define sub(a, b) ((a) < (b) ? (b) - (a) : (a) - (b))
template <class T, class POW> void fastpow(T x, POW n, POW p, T &ans) {
  for (; n; n >>= 1) {
    if (n & 1) {
      ans *= x;
      ans %= p;
    }
    x *= x;
    x %= p;
  }
}
/*input x, n, p, ans, will modify ans to x ^ n % p
the first is x, ans and the second is n, p (LL or __int128)
*/
uLL pri[7] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022}; /*2^64*/
// int p[3]={2,7,61};/*2^32*/
bool check(const uLL x, const uLL p) {
  uLL d = x - 1, ans = 1;
  fastpow(p, d, x, ans);
  if (ans != 1)
    return 1;
  for (; !(d & 1);) {
    d >>= 1;
    ans = 1;
    fastpow(p, d, x, ans);
    if (ans == x - 1)
      return 0;
    else if (ans != 1)
      return 1;
  }
  return 0;
}
bool miller_rabin(const uLL x) {
  if (x == 1)
    return 0;
  for (auto e : pri) {
    if (e >= x)
      return 1;
    if (check(x, e))
      return 0;
  }
  return 1;
}
template <class T> T gcd(T a, T b) {
  if (!a)
    return b;
  if (!b)
    return a;
  if (a & b & 1)
    return gcd(sub(a, b), min(a, b));
  if (a & 1)
    return gcd(a, b >> 1);
  if (b & 1)
    return gcd(a >> 1, b);
  return gcd(a >> 1, b >> 1) << 1;
}
/*gcd(a,b) denote gcd(a, 0) = a*/
mt19937 rnd(time(0));
template <class T> T f(T x, T c, T mod) {
  return (((uLL)x) * x % mod + c) % mod;
}
template <class T> T rho(T n) {
  T mod = n, x = rnd() % mod, c = rnd() % (mod - 1) + 1, p = 1;
  for (T i = 2, j = 2, d = x;; ++i) {
    x = f(x, c, mod);
    p = ((uLL)p) * sub(x, d) % mod;
    if (i % 127 == 0 && gcd(p, n) != 1)
      return gcd(p, n);
    if (i == j) {
      j <<= 1, d = x;
      if (gcd(p, n) != 1)
        return gcd(p, n);
    }
  }
}
template <class T> T pollard_rho(T n) {
  if (miller_rabin(n))
    return n;
  T p = n;
  while (p == n)
    p = rho(n);
  return max(pollard_rho(p), pollard_rho(n / p));
}
int main() {
  LL t, n, ans;
  for (cin >> t; t--;) {
    cin >> n;
    ans = pollard_rho(n);
    if (ans == n)
      puts("Prime");
    else
      printf("%lld\n", ans);
  }
}
```

2.11. Formula

2.11.1. Dirichlet Convolution

$\varepsilon = \mu * 1$
$\varphi = \mu * \mathrm{Id}$

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

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

2.11.8. ext-Kummer Theorem

$v(\binom{n}{m1,m2,...mk}) = \frac{\sum_{i=1}^{k} s(mi)-s(n)}{p-1}$

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 $\binom{n_i}{m_i}\%p = 0$. 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 <=> 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

```

#define int long long
using namespace std;

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

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

template <class T> class Matrix {
  static const T MOD = 1000000007;

public:
  vector<vector<T>> v;
  Matrix(int n, int m, int identity) {
    v = vector<vector<T>>(n, vector<T>(m, 0));
    if (identity)
      for (int i = 0, k = min(n, m); i < k; ++i)
        v[i][i] = 1;
  }
  Matrix(Matrix &b) { v = b.v; }
  void in(int l = 0, int m = -1, int u = 0, int n = -1) {
    if (n < 0)
      n = v.size();
    if (m < 0)
      m = v[0].size();
    for (int i = u; i < n; ++i)
      for (int j = l; j < m; ++j)
        scanf("%lld", &v[i][j]);
  }
  Matrix(int n, int m) {
    v = vector<vector<T>>(n, vector<T>(m, 0));
    in();
  }
  void out(int l = 0, int m = -1, int u = 0, int n = -1) {
    if (n < 0)
      n = v.size();
    if (m < 0)
      m = v[0].size();
    for (int i = u; i < n; ++i)
      for (int j = l; j < m; ++j)
        printf("%lld%c", v[i][j], " \n"[j == m - 1]);
  }
  Matrix operator=(Matrix &b) {
    v = b.v;
    return *this;
  }
  Matrix operator+(Matrix &b) {
    Matrix ans(*this);
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
      for (int j = 0; j < m; ++j) {
        ans.v[i][j] += b.v[i][j];
        if (MOD) {
          if (ans.v[i][j] < 0)
            ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
          if (ans.v[i][j] >= MOD)
            ans.v[i][j] %= MOD;
        }
      }
    return ans;
  }
  Matrix operator+(T x) {
    Matrix ans(*this);
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
      for (int j = 0; j < m; ++j) {
        ans.v[i][j] += x;
        if (MOD) {
          if (ans.v[i][j] < 0)
            ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
          if (ans.v[i][j] >= MOD)
            ans.v[i][j] %= MOD;
        }
      }
    return ans;
  }
  Matrix operator-(Matrix &b) {
    Matrix ans(*this);
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
      for (int j = 0; j < m; ++j) {
        ans.v[i][j] -= b.v[i][j];
        if (MOD) {
          if (ans.v[i][j] < 0)
            ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
          if (ans.v[i][j] >= MOD)
            ans.v[i][j] %= MOD;
        }
      }
    return ans;
  }
  Matrix operator-(T x) {
    Matrix ans(*this);
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
      for (int j = 0; j < m; ++j) {
        ans.v[i][j] -= x;
        if (MOD) {
          if (ans.v[i][j] < 0)
            ans.v[i][j] = (ans.v[i][j] % MOD + MOD) % MOD;
          if (ans.v[i][j] >= MOD)
            ans.v[i][j] %= MOD;
        }
      }
    return ans;
  }
  Matrix operator+=(Matrix &b) {
    int n = v.size(), m = v[0].size();
    for (int i = 0; i < n; ++i)
      for (int j = 0; j < m; ++j) {
        v[i][j] += b.v[i][j];
        if (MOD) {
          if (v[i][j] < 0)
            v[i][j] = (v[i][j] % MOD + MOD) % MOD;
          if (v[i][j] >= MOD)
            v[i][j] %= MOD;
        }
      }
    return *this;
```

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

```
            v[i][k] -= v[curi][k] * v[i][j];
            if (MOD) {
              v[i][k] = (v[i][k] % MOD + MOD) % MOD;
            }
          }
        }
      }
      curi++;
    }
  }
};
```

3. String

3.1. KMP

```
string s, t;
int pmt[1000005];

void init() {
  for (int i = 1, j = 0; i < t.size(); i++) {
    while (j && t[j] ^ t[i]) {
      j = pmt[j - 1];
    }
    if (t[j] == t[i])
      j++;
    pmt[i] = j;
  }
}

int kmp(string s) {
  int ret = 0;
  for (int i = 0, j = 0; i < s.size(); i++) {
    while (j && s[i] ^ t[j]) {
      j = pmt[j - 1];
    }
    if (s[i] == t[j]) {
      j++;
    }
    if (j == t.size()) {
      ret++;
      j = pmt[j - 1];
    }
  }
  return ret;
}
```

3.2. Longest Palindrome

```

#define int long long
using namespace std;

string s;
string t;
int n;
int d[2000005];
int ans = 0;

signed main() {
  cin >> t;
  n = t.size();
  for (int i = 0; i < 2 * n + 1; i++) {
    if (i & 1 ^ 1) {
      s += '0';
    } else {
      s += t[i / 2];
    }
  }
  n = s.size();
  d[0] = 1;
  for (int i = 0, l = 0, r = 0; i < n; i++) {
    if (i > r) {
      d[i] = 1;
      bool a = i + d[i] < n;
      bool b = i - d[i] >= 0;
      bool c = (s[i + d[i]] == s[i - d[i]];
      while (a && b && c) {
        d[i]++;
        a = i + d[i] < n;
        b = i - d[i] >= 0;
        c = ([i + d[i]] == s[i - d[i]]);
      }
      l = i - d[i] + 1;
      r = i + d[i] - 1;
    } else {
      int j = l + r - i;
      if (j - d[j] + 1 > l) {
        d[i] = d[j];
      } else {
        d[i] = r - i + 1;
        a = i + d[i] < n;
        b = i - d[i] >= 0;
        c = (s[i + d[i]] == s[i - d[i]]);
        while (a && b && c) {
          d[i]++;
          a = i + d[i] < n;
          b = i - d[i] >= 0;
          c = (s[i + d[i]] == s[i - d[i]]);
        }
        l = i - d[i] + 1;
        r = i + d[i] - 1;
      }
    }
    // cout << d[i] << " ";
    if (d[i] > d[ans]) {
      ans = i;
    }
  }
  for (int i = ans - d[ans] + 1; i < ans + d[ans]; i++) {
    if (s[i] ^ '0') {
      cout << s[i];
    }
  }
}
```

3.3. Z

```

#define int long long
using namespace std;

string s, t;
int ans = 0;

int z[2000005];

signed main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  cin >> s >> t;
  s = t + '0' + s;
  int n, m;
  n = s.size();
  m = t.size();
  for (int i = 0, l = 0, r = 0; i < n; i++) {
    if (z[i - l] < r - i + 1) {
      z[i] = z[i - l];
    } else {
      z[i] = max(r - i + 1, (int)0);
      while (i + z[i] < n && s[i + z[i]] == s[z[i]]) {
        z[i]++;
      }
      l = i;
      r = i + z[i] - 1;
      if (z[i] == m) {
        ans++;
      }
    }
  }
  cout << ans;
}
```

4. Graph

4.1. one-out-degree (CSES Planets Cycles)

```

#define int long long
using namespace std;

int n, q;
int a[200005];
int r[200005];
int d[200005];
int cycle[200005];
int len[200005];
int cnt = 0;
vector<int> v[200005];
bitset<200005> vis1;
bitset<200005> vis2;

void findcycle(int x) {
  while (!vis1[x]) {
    vis1[x] = 1;
    x = a[x];
  }
  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

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

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

```

#define int long long
using namespace std;

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

bool bfs() {
  memset(head, 0, sizeof(head));
  memset(lv, 0, sizeof(lv));
  queue<int> q;
  q.push(1);
  while (!q.empty()) {
    int now = q.front();
    q.pop();
    if (now == n)
      continue;
    for (int i : v[now]) {
      if (i != 1 && c[now][i] && !lv[i]) {
        lv[i] = lv[now] + 1;
        q.push(i);
      }
    }
  }
  return lv[n];
}

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

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

4.4. SCC

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

void dfs(int x) {
  d[x] = low[x] = ++cnt;
  s.push(x);
  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]);
    }
  }
  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)

```

#define int long long
using namespace std;

int n, m;
vector<int> v[200005];
int d[200005];
int low[200005];
int cnt = 0;
int now = 0;
int scc[200005];
stack<int> s;
int op[200005];
vector<int> v2[200005];
int ind[200005];
queue<int> q;
int ans[200005];

int no(int x) {
  if (x > m)
    return x - m;
  return x + m;
}

void dfs(int x) {
  d[x] = low[x] = ++cnt;
  s.push(x);
  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]);
    }
  }
  if (d[x] == low[x]) {
    now++;
    while (!s.empty()) {
      int k = s.top();
      s.pop();
      scc[k] = now;
      if (k == x)
        break;
    }
  }
}

signed main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  cin >> n >> m;
  while (n--) {
    char a, b;
    int x, y;
    cin >> a >> x >> b >> y;
    if (a == '-')
      x = no(x);
    if (b == '-')
      y = no(y);
    v[no(x)].push_back(y);
    v[no(y)].push_back(x);
  }
  for (int i = 1; i <= 2 * m; i++) {
    if (!d[i]) {
      dfs(i);
    }
  }
  for (int i = 1; i <= m; i++) {
    if (scc[i] ^ scc[i + m]) {
      op[scc[i]] = scc[i + m];
      op[scc[i + m]] = scc[i];
    } else {
      cout << "IMPOSSIBLE";
      exit(0);
    }
  }
  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

```
struct line {
  int a, b = 1000000000000000;
  int y(int x) { return a * x + b; }
};

line tree[4000005];
int n, x;
int s[200005];
int f[200005];
int dp[200005];

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

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

5.2. CHO

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

```
};

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

6. Geometry

6.1. Intersect

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

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

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

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

6.2. Inside

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

6.3. Minimum Euclidean Distance

```

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

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

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

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

6.4. Convex Hull

```

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

using namespace std;

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

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

template <typename T> vector<pair<T, T>> getCH(vector<pair<T, T>> v
  int n = v.size();
  sort(v.begin(), v.end());
  vector<pair<T, T>> hull;
  for (int i = 0; i < 2; i++) {
    int t = hull.size();
    for (auto x : v) {
      while (hull.size() - t >= 2 &&
             cross(hull[hull.size() - 1] - hull[hull.size() - 2],
                   x - hull[hull.size() - 2]) <= 0)
        hull.pop_back();
      hull.push_back(x);
    }
    hull.pop_back();
    reverse(v.begin(), v.end());
  }
  return hull;
}
```

7. Tree

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

```

#define int long long
```

```cpp
using namespace std;

int tree[800005];

int n, q;
int a[200005];
int st[200005];
int tp[200005];
int p[200005];
int cnt = 0;
int d[200005];
int si[200005];
vector<int> v[200005];
int b[200005];

void build(int l = 1, int r = n, int index = 1) {
  if (l == r) {
    tree[index] = b[l];
    return;
  }
  int mid = (l + r) >> 1;
  build(l, mid, index << 1);
  build(mid + 1, r, index << 1 | 1);
  tree[index] = max(tree[index << 1], tree[index << 1 | 1]);
}

int query(int L, int R, int l = 1, int r = n, int index = 1) {
  if (L == l && r == R) {
    return tree[index];
  }
  int mid = (l + r) >> 1;
  if (R <= mid) {
    return query(L, R, l, mid, index << 1);
  }
  if (L > mid) {
    return query(L, R, mid + 1, r, index << 1 | 1);
  }
  return max(query(L, mid, l, mid, index << 1),
             query(mid + 1, R, mid + 1, r, index << 1 | 1));
}

void modify(int x, int val, int l = 1, int r = n, int index = 1) {
  if (l == r) {
    tree[index] = val;
    return;
  }
  int mid = (l + r) >> 1;
  if (x <= mid) {
    modify(x, val, l, mid, index << 1);
  } else {
    modify(x, val, mid + 1, r, index << 1 | 1);
  }
  tree[index] = max(tree[index << 1], tree[index << 1 | 1]);
}

void dfs(int x, int pre) {
  si[x] = 1;
  for (int i : v[x]) {
    if (i == pre)
      continue;
    p[i] = x;
    d[i] = d[x] + 1;
    dfs(i, x);
    si[x] += si[i];
  }
}

void dfs2(int x, int pre, int t) {
  tp[x] = t;
  st[x] = ++cnt;
  int ma = 0;
  for (int i : v[x]) {
    if (i == pre)
      continue;
    if (si[i] > si[ma]) {
      ma = i;
    }
  }
  if (!ma)
    return;
  dfs2(ma, x, t);
  for (int i : v[x]) {
    if (i == pre || i == ma) {
      continue;
    }
    dfs2(i, x, i);
  }
}

int f(int x, int y) {
  int ret = 0;
  while (tp[x] ^ tp[y]) {
    if (d[tp[x]] < d[tp[y]]) {
      swap(x, y);
    }
    ret = max(ret, query(st[tp[x]], st[x]));
    x = p[tp[x]];
  }
  if (d[x] > d[y])
    swap(x, y);
  ret = max(ret, query(st[x], st[y]));
  return ret;
}

signed main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  cin >> n >> q;
  for (int i = 1; i <= n; i++) {
    cin >> a[i];
  }
  for (int i = 1; i < n; i++) {
    int x, y;
    cin >> x >> y;
    v[x].push_back(y);
    v[y].push_back(x);
  }
  dfs(1, 0);
  dfs2(1, 0, 1);
  for (int i = 1; i <= n; i++) {
    b[st[i]] = a[i];
  }
  build();
  while (q--) {
    int mode, x, y;
    cin >> mode >> x >> y;
    if (mode == 1) {
      modify(st[x], y);
    } else {
      cout << f(x, y) << " ";
    }
  }
}
```

7.2. LCA

```cpp

#define int long long
using namespace std;

int n, q;
int a[200005][21];
int d[200005];
vector<int> v[200005];

void init() {
  for (int j = 1; j < 21; j++) {
    for (int i = 1; i <= n; i++) {
      a[i][j] = a[a[i][j - 1]][j - 1];
    }
  }
}

void dfs(int x, int pre) {
  for (int i : v[x]) {
    if (i == pre) {
      continue;
    }
    a[i][0] = x;
    d[i] = d[x] + 1;
    dfs(i, x);
  }
}

int lca(int x, int y) {
  while (d[x] ^ d[y]) {
    if (d[x] < d[y]) {
      swap(x, y);
    }
    int k = __lg(d[x] - d[y]);
    x = a[x][k];
  }
  if (x == y) {
    return x;
  }
  for (int i = 20; i >= 0; i--) {
    if (a[x][i] != a[y][i]) {
      x = a[x][i];
      y = a[y][i];
```

```
    }
  }
  return a[x][0];
}

signed main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  cin >> n >> q;
  for (int i = 1; i < n; i++) {
    int x, y;
    cin >> x >> y;
    v[x].push_back(y);
    v[y].push_back(x);
  }
  dfs(1, 0);
  init();
  while (q--) {
    int x, y;
    cin >> x >> y;
    int k = lca(x, y);
    cout << (d[x] + d[y] - 2 * d[k]) << "\n";
  }
}
```

8. Misc

8.1. Tri Search

```

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

double get(double x) {
  double ret = 0;
  double k = 1;
  for (int i = 0; i <= n; i++) {
    ret += k * a[i];
    k *= x;
  }
  return -ret;
}

template <class T> T bi_search(T l, T r, T end) {
  if (!check(r - end))
    return r - end;
  for (; r - l > end;) {
    T mid = (l + r) / 2;
    if (check(mid))
      r = mid;
    else
      l = mid;
  }
  return l;
}
/*check gives 000000001111 find the last 0*/

template <class T> T tri_search(T l, T r, T end) {
  T midl, midr;
  for (;;) {
    midl = (l + r) / 2;
    midr = (midl + r) / 2;
    if (midr - midl < end)
      break;
    if (get(midr) > get(midl))
      r = midr;
    else
      l = midl;
  }
  for (; r - l > end;) {
    midl = (l + r) / 2;
    if (get(r) > get(l))
      r = midl;
    else
      l = midl;
  }
  return l;
}
/*get gives the value, find the minimum*/

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