NTJ
UDESC

Eric Grochowicz, Enzo de Almeida Rodrigues e João Marcos de Oliveira

29 de agosto de 2023

Índice

1 Strings

1.1 Automato de Aho Corasick

```
// Fonte: https://github.com/shahjalalshohag/code-library
//
// Faz coisarada

const int N = 3e5 + 5;

struct AC {
    int N, P;
    const int A = 26;
    vector<vector<int>> next;
    vector<int> link, out_link;
    vector<vector<int>> out;
    AC() : N(0), P(0) { node(); }
    int node() {
        next.emplace_back(A, 0);
        link.emplace_back(0);
        out_link.emplace_back(0);
        out.emplace_back(0);
        return N++;
    }
    inline int get(char c) { return c - 'a'; }
    int add_pattern(const string T) {
        int u = 0;
        for (auto c : T) {
            if (!next[u][get(c)]) next[u][get(c)] = node();
            u = next[u][get(c)];
        }
        out[u].push_back(P);
        return P++;
    }
    void compute() {
        queue<int> q;
        for (q.push(0); !q.empty();) {
            int u = q.front();
            q.pop();
            for (int c = 0; c < A; ++c) {
                int v = next[u][c];
                if (!v)
                    next[u][c] = next[link[u]][c];
                else {
                    link[v] = u ? next[link[u]][c] : 0;
                    out_link[v] =
                        out[link[v]].empty() ? out_link[link[v]] : link[v];
                    q.push(v);
                }
            }
        }
    }
    int advance(int u, char c) {
        while (u && !next[u][get(c)]) u = link[u];
        u = next[u][get(c)];
        return u;
    }
};

int32_t main() {
```

```
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    auto st = clock();
    int t, cs = 0;
    cin >> t;
    while (t--) {
        int n;
        cin >> n;
        vector<string> v;
        for (int i = 0; i < n; i++) {
            string s;
            cin >> s;
            v.push_back(s);
        }
        sort(v.begin(), v.end());
        v.erase(unique(v.begin(), v.end()), v.end());
        AC aho;
        vector<int> len(n + 3, 0);
        for (auto s : v) {
            len[aho.add_pattern(s)] = s.size();
        }
        aho.compute();
        string s;
        cin >> s;
        n = s.size();
        vector<int> dp(n, n + 10);
        int u = 0;
        for (int i = 0; i < n; i++) {
            char c = s[i];
            u = aho.advance(u, c);
            for (int v = u; v; v = aho.out_link[v]) {
                for (auto p : aho.out[v]) {
                    dp[i] =
                        min(dp[i], (i - len[p] >= 0 ? dp[i - len[p]] : 0) + 1);
                }
            }
        }
        cout << "Case " << ++cs << ": ";
        if (dp[n - 1] == n + 10) {
            cout << "impossible\n";
        } else {
            cout << dp[n - 1] << '\n';
        }
    }
    cout << 1.0 * (clock() - st) / 1000 << '\n';
    return 0;
}
```

1.2 Hashing

```
// Hashing estatico
// Tambem funciona com vector
//
// Build: O(n)
// Query (operator): O(1)

const int mods = 1;
typedef array<ll, mods> Hash; // array<T, mods>
```

```
template <typename obj = string, typename T = ll, typename U = __int128_t>
struct Hashing {
   int N;
   bool inverse = 0;
   U p = 1001003;
   Hash mod{(T)1e18 + 9};
   array<vector<T>, mods> pw, hsh;
   void build(obj s, bool _inverse = 0) {
       inverse = _inverse;
       if (inverse) reverse(begin(s), end(s));
       N = size(s);
       for (int j = 0; j < mods; j++) {
           pw[j].resize(N), hsh[j].resize(N);
           pw[j][0] = 1;
           hsh[j][0] = s[0] % mod[j];
           for (int i = 1; i < N; i++) {
               pw[j][i] = (U)pw[j][i - 1] * p % mod[j];
               hsh[j][i] = ((U)hsh[j][i - 1] * p + s[i]) % mod[j];
           }
       }
   }
   Hash operator()(int l, int r) {
       if (inverse) { l = N - 1 - l, r = N - 1 - r; swap(l, r); }
       Hash ans;
       for (int j = 0; j < mods; j++) {
           ans[j] = hsh[j][r];
           if (l > 0) {
               ans[j] = (ans[j] - ((U)hsh[j][l - 1] * pw[j][r - l + 1] %
                   mod[j]) + mod[j]) % mod[j];
           }
       }
       return ans;
   }
};

// Hashing dinamico
//
// Hash que usa uma Fenwick pra updatar
//
// Build: O(n)
// Query (operator): O(log(n))
// Update: O(log(n))

const int mods = 1;
typedef array<ll, mods> Hash; // array<T, mods>

template <typename obj = string, typename T = ll, typename U = __int128_t>
struct Hashing {
   int N;
   bool inverse = 0;
   U p = 1001003;
   Hash mod{(ll)1e18 + 9};
   array<vector<ll>, mods> pw, hsh, inv;
   void build(obj s, bool _inverse = 0) {
       inverse = _inverse;
       if (inverse) reverse(begin(s), end(s));
       N = size(s);
       for (int j = 0; j < mods; j++) {
           pw[j].resize(N + 1), hsh[j].resize(N + 1), inv[j].resize(N + 1);
           pw[j][1] = p;
           pw[j][0] = inv[j][0] = inv[j][1] = 1;
```

```
            U b = p;
            for (U e = mod[j] - 2; e > 0; e >>= 1, b = b * b % mod[j]) {
                if (e & 1) inv[j][1] = (U)inv[j][1] * b % mod[j];
            }
            for (int i = 2; i <= N; i++) {
                inv[j][i] = (U)inv[j][i - 1] * inv[j][1] % mod[j];
                pw[j][i] = (U)pw[j][i - 1] * pw[j][1] % mod[j];
            }
            hsh[j][1] = s[0] * pw[j][1] % mod[j];
            for (int i = 2; i <= N; i++) {
                hsh[j][i] = (U)s[i - 1] * pw[j][i] % mod[j];
            }
            for (int i = 1; i <= N; i++) {
                int u = i + (i & -i);
                if (u <= N) hsh[j][u] = (hsh[j][u] + hsh[j][i]) % mod[j];
            }
        }
    }
    Hash operator()(int l, int r) {
        assert(l <= r);
        if (inverse) { l = N - 1 - l, r = N - 1 - r; swap(l, r); }
        Hash ans;
        l += 1, r += 1;
        for (int j = 0; j < mods; j++) {
            ans[j] = 0;
            for (int i = r; i > 0; i -= (i & -i)) {
                ans[j] = (ans[j] + hsh[j][i]) % mod[j];
            }
            for (int i = l - 1; i > 0; i -= (i & -i)) {
                ans[j] = (ans[j] - hsh[j][i] + mod[j]) % mod[j];
            }
            ans[j] = (U)ans[j] * inv[j][l - 1] % mod[j];
        }
        return ans;
    }
    void update(int idx, T val) {
        if (inverse) idx = N - 1 - idx;
        idx += 1;
        for (int j = 0; j < mods; j++) {
            T u = 0;
            for (int i = idx; i > 0; i -= (i & -i)) {
                u = (u + hsh[j][i]) % mod[j];
            }
            for (int i = idx - 1; i > 0; i -= (i & -i)) {
                u = (u - hsh[j][i] + mod[j]) % mod[j];
            }
            for (int i = idx; i <= N; i += (i & -i)) {
                hsh[j][i] = (hsh[j][i] - u + mod[j]) % mod[j];
            }
            for (int i = idx; i <= N; i += (i & -i)) {
                hsh[j][i] = (hsh[j][i] + (U)val * pw[j][idx] % mod[j]) %
                    mod[j];
            }
        }
    }
};
```

2 Problemas

2.1 Kth digito na string infinita de digitos

```
// Retorna qual o numero e qual o algarismo do Kth digito
// na string infinita dos numeros naturais (12345678910111213...)
// Complexidade: O(log_10(k))

pair<ll, ll> kthdig(ll k) {
    ll qtd = 1, num_alg = 1, base = 1;
    while (1) {
        ll add = (9 * base) * num_alg;
        if (qtd + add < k) {
            qtd += add;
        } else break;
        base *= 10, num_alg++;
    }
    ll algarismo = (k - qtd) % num_alg;
    ll numero = (k - qtd) / num_alg + base;
    return {numero, algarismo};
}
```

3 Estruturas

3.1 Fenwick Tree

```
// Processas queries de operacao com inverso
//
// Build: O(n)
// Query: O(log(n))
// Update: O(log(n))

typedef long long ll;

struct fenwick {
    vector<ll> bit;
    fenwick(int n) { bit.assign(n+1, 0); }
    fenwick(vector<ll>& v) {
        int n = v.size();
        bit.assign(n+1, 0);
        for(int i = 1; i <= n; i++) bit[i] = v[i-1];
        for(int i = 1; i <= n; i++) {
            int j = i + (i & -i);
            if(j <= n) bit[j] += bit[i];
        }
    }
    ll query(int i){
        ll res = 0;
        for(; i; i -= (i & -i))
            res += bit[i];
        return res;
    }
    ll query(int l, int r){
        return query(r) - query(l-1);
    }
    void update(int i, ll d){
        for(; i && i < (int)bit.size(); i += (i & -i))
            bit[i] += d;
    }
};
```

3.2 Segment Tree Beats

```
// Fonte: https://usaco.guide/adv/segtree-beats
//
// Faz coisarada

const int MAXN = 3e5 + 5;

int N, Q;
long long tsum[MAXN * 4], tmax[MAXN * 4];

void update_mod(int l, int r, long long v, int t = 1, int tl = 1, int tr = N)
   {
   if (r < tl || tr < l || tmax[t] < v) {
       return;
   } else if (tl == tr) {
       int val = tmax[t] % v;
       tsum[t] = tmax[t] = val;
       return;
   }
```

```
    int tm = (tl + tr) / 2;
    update_mod(l, r, v, t * 2, tl, tm);
    update_mod(l, r, v, t * 2 + 1, tm + 1, tr);
    tsum[t] = tsum[t * 2] + tsum[t * 2 + 1];
    tmax[t] = max(tmax[t * 2], tmax[t * 2 + 1]);
}

void update_set(int i, long long v, int t = 1, int tl = 1, int tr = N) {
    if (tl == tr) {
        tsum[t] = tmax[t] = v;
        return;
    }

    int tm = (tl + tr) / 2;
    if (i <= tm) {
        update_set(i, v, t * 2, tl, tm);
    } else {
        update_set(i, v, t * 2 + 1, tm + 1, tr);
    }
    tsum[t] = tsum[t * 2] + tsum[t * 2 + 1];
    tmax[t] = max(tmax[t * 2], tmax[t * 2 + 1]);
}

long long query(int l, int r, int t = 1, int tl = 1, int tr = N) {
    if (r < tl || tr < l) {
        return 0;
    } else if (l <= tl && tr <= r) {
        return tsum[t];
    }

    int tm = (tl + tr) / 2;
    return query(l, r, t * 2, tl, tm) + query(l, r, t * 2 + 1, tm + 1, tr);
}

int main() {
    cin >> N >> Q;
    for (int i = 1; i <= N; i++) {
        long long a;
        cin >> a;
        update_set(i, a);
    }
    for (int q = 0; q < Q; q++) {
        int t;
        cin >> t;
        if (t == 1) {
            int l, r;
            cin >> l >> r;
            cout << query(l, r) << '\n';
        } else if (t == 2) {
            int l, r;
            long long x;
            cin >> l >> r >> x;
            update_mod(l, r, x);
        } else if (t == 3) {
            int i;
            long long x;
            cin >> i >> x;
            update_set(i, x);
        }
    }
```

```
}
```

4 Grafos

4.1 Binary Lifting

```
// Binary Lifting (em nodos)
//
// Computa LCA e tambem resolve queries de operacoes
// associativas e comutativas em caminhos.
//
// Build(): O(n log(n))
// Query(): O(log(n))
// Lca(): O(log(n))
//
// up[u][i] = (2 ^ i)-esimo pai do u
// st[u][i] = query ate (2 ^ i)-esimo pai do u (NAO INCLUI O U)

 const int maxn = 3e5 + 5, LG = 20;
 vector<int> adj[maxn];

 struct BinaryLifting {
     int up[maxn][LG], st[maxn][LG], val[maxn], t = 1;
     int tin[maxn], tout[maxn];

     const int neutral = 0;
     int merge(int l, int r) { return l + r; }

     void build(int u, int p = -1) {
         tin[u] = t++;
         for (int i = 0; i < LG - 1; i++) {
             up[u][i + 1] = up[up[u][i]][i];
             st[u][i + 1] = merge(st[u][i], st[up[u][i]][i]);
         }
         for (int v : adj[u]) if (v != p) {
             up[v][0] = u, st[v][0] = val[u];
             build(v, u);
         }
         tout[u] = t++;
     }

     void build(int root, vector<int> &v) {
         t = 1;
         int N = size(v);
         for (int i = 0; i < N; i++) val[i] = v[i];
         up[root][0] = root;
         st[root][0] = val[root];
         build(root);
     }

     bool ancestor(int u, int v) {
         return tin[u] <= tin[v] && tout[u] >= tout[v];
     }

     int query2(int u, int v, bool include_lca) {
         if (ancestor(u, v)) return include_lca ? val[u] : neutral;
         int ans = val[u];
         for (int i = LG - 1; i >= 0; i--) {
             if (!ancestor(up[u][i], v)) {
                 ans = merge(ans, st[u][i]);
                 u = up[u][i];
             }
```

```
        }
        return include_lca ? merge(ans, st[u][0]) : ans;
    }

    int query(int u, int v) {
        if (u == v) return val[u];
        return merge(query2(u, v, 1), query2(v, u, 0));
    }

    int lca(int u, int v) {
        if (ancestor(u, v)) return u;
        if (ancestor(v, u)) return v;
        for (int i = LG - 1; i >= 0; i--) {
            if (!ancestor(up[u][i], v)) {
                u = up[u][i];
            }
        }
        return up[u][0];
    }

} bl;

// Binary Lifting (em arestas)
//
// up[u][i] = (2 ^ i)-esimo pai do u
// st[u][i] = query ate (2 ^ i)-esimo pai do u

const int maxn = 3e5 + 5, LG = 20;
vector<pair<int, int>> adj[maxn];

struct BinaryLifting {
    int up[maxn][LG], st[maxn][LG], t = 1;
    int tin[maxn], tout[maxn];

    const int neutral = 0;
    int merge(int l, int r) { return l + r; }

    void build(int u, int p = -1) {
        tin[u] = t++;
        for (int i = 0; i < LG - 1; i++) {
            up[u][i + 1] = up[up[u][i]][i];
            st[u][i + 1] = merge(st[u][i], st[up[u][i]][i]);
        }
        for (auto [w, v] : adj[u]) if (v != p) {
            up[v][0] = u, st[v][0] = w;
            build(v, u);
        }
        tout[u] = t++;
    }

    bool ancestor(int u, int v) {
        return tin[u] <= tin[v] && tout[u] >= tout[v];
    }

    int query2(int u, int v) {
        if (ancestor(u, v)) return neutral;
        int ans = neutral;
        for (int i = LG - 1; i >= 0; i--) {
            if (!ancestor(up[u][i], v)) {
                ans = merge(ans, st[u][i]);
                u = up[u][i];
```

```
            }
        }
        return merge(ans, st[u][0]);
    }

    int query(int u, int v) {
        if (u == v) return neutral;
#warning TRATAR ESSE CASO ACIMA
        return merge(query2(u, v), query2(v, u));
    }
} bl;

// Binary Lifting para operacoes nao comutativas (em nodos)
//
// Levemente diferente do padrao
//
// Esse aqui resolve query de Kadani em arvore
// https://codeforces.com/contest/1843/problem/F2

const int maxn = 3e5 + 5, LG = 20;
vector<int> adj[maxn];

struct node {
    int pref, suff, sum, best;
};
const node neutral = {0, 0, 0, 0};
node new_node(int x) {
    return node{x, x, x, x};
}

node merge(node& l, node& r) {
    int pref = max(l.pref, l.sum + r.pref);
    int suff = max(r.suff, r.sum + l.suff);
    int sum = l.sum + r.sum;
    int best = max(l.suff + r.pref, max(l.best, r.best));
    return node{pref, suff, sum, best};
}

struct BinaryLifting {
    int up[maxn][LG], val[maxn], t = 1;
    int tin[maxn], tout[maxn];
    node st[maxn][LG], st2[maxn][LG];

    void build(int u, int p = -1) {
        tin[u] = t++;
        for (int i = 0; i < LG - 1; i++) {
            up[u][i + 1] = up[up[u][i]][i];
            st[u][i + 1] = merge(st[u][i], st[up[u][i]][i]);
            st2[u][i + 1] = merge(st2[up[u][i]][i], st2[u][i]);
        }
        for (int v : adj[u]) if (v != p) {
            up[v][0] = u;
            st[v][0] = new_node(val[u]);
            st2[v][0] = new_node(val[u]);
            build(v, u);
        }
        tout[u] = t++;
    }

    void build(int root, vector<int> &v) {
        t = 1;
```

```
        int N = size(v);
        for (int i = 0; i < N; i++) val[i] = v[i];
        up[root][0] = root;
        st[root][0] = new_node(val[root]);
        st2[root][0] = new_node(val[u]);
        build(root);
    }

    bool ancestor(int u, int v) {
        return tin[u] <= tin[v] && tout[u] >= tout[v];
    }

    node query2(int u, int v, bool include_lca, bool invert) {
        if (ancestor(u, v)) return include_lca ? new_node(val[u]) : neutral;
        node ans = new_node(val[u]);
        for (int i = LG - 1; i >= 0; i--) {
            if (!ancestor(up[u][i], v)) {
                if (invert) ans = merge(st2[u][i], ans);
                else ans = merge(ans, st[u][i]);
                u = up[u][i];
            }
        }
        if (!include_lca) return ans;
        return merge(ans, st[u][0]);
    }

    node query(int u, int v) {
        if (u == v) return new_node(val[u]);
        node l = query2(u, v, 1, 0);
        node r = query2(v, u, 0, 1);
        return merge(l, r);
    }

    int lca(int u, int v) {
        if (ancestor(u, v)) return u;
        if (ancestor(v, u)) return v;
        for (int i = LG - 1; i >= 0; i--) {
            if (!ancestor(up[u][i], v)) {
                u = up[u][i];
            }
        }
        return up[u][0];
    }

} bl, bl2;
```

4.2 Bridges e Edge Biconnected Components

```
// Acha todas as pontes em O(n)
// Tambem constroi a arvore condensada, mantendo
// so as pontes como arestas e o resto comprimindo
// em nodos
//
// Salva no vetor bridges os pares {u, v} cujas arestas sao pontes

typedef pair<int, int> ii;
const int maxn = 2e5 + 5;
int n, m;
bool vis[maxn];
```

```
int dp[maxn], dep[maxn];
vector<int> adj[maxn];
vector<ii> bridges;

void dfs_dp(int u, int p = -1, int d = 0) {
    dp[u] = 0, dep[u] = d, vis[u] = 1;
    for (auto v : adj[u]) if (v != p) {
        if (vis[v]){
            if (dep[v] < dep[u]) dp[v]--, dp[u]++;
        } else {
            dfs_dp(v, u, d + 1);
            dp[u] += dp[v];
        }
    }
    if (dp[u] == 0 && p != -1) { // edge {u, p} eh uma ponte
        bridges.emplace_back(u, p);
    }
}

void find_bridges() {
    memset(vis, 0, n);
    for (int i = 0; i < n; i++) if (!vis[i]) {
        dfs_dp(i);
    }
}

// Edge Biconnected Components (requer todo codigo acima)

int ebcc[maxn], ncc = 0;
vector<int> adjbcc[maxn];

void dfs_ebcc(int u, int p, int cc) {
    vis[u] = 1;
    if (dp[u] == 0 && p != -1) {
        cc = ++ncc;
    }
    ebcc[u] = cc;
    for (auto v : adj[u]) if (!vis[v]) {
        dfs_ebcc(v, u, cc);
    }
}

void build_ebcc_graph() {
    find_bridges();
    memset(vis, 0, n);
    for (int i = 0; i < n; i++) if (!vis[i]) {
        dfs_ebcc(i, -1, ncc);
        ++ncc;
    }
    // Opcao 1 - constroi o grafo condensado passando por todas as edges
    for (int u = 0; u < n; u++) {
        for (auto v : adj[u]) {
            if (ebcc[u] != ebcc[v]) {
                adjbcc[ebcc[u]].emplace_back(ebcc[v]);
            } else {
                // faz algo
            }
        }
    }
    // Opcao 2 - constroi o grafo condensado passando so pelas pontes
    for (auto [u, v] : bridges) {
```

```
            adjbcc[ebcc[u]].emplace_back(ebcc[v]);
            adjbcc[ebcc[v]].emplace_back(ebcc[u]);
        }
    }
```

4.3 Pontos de articulacao

```
// Fonte: https://github.com/shahjalalshohag/code-library
//
// O equivalente a pontes, em vertices
//
// Complexidade: O(n)

#include<bits/stdc++.h>
using namespace std;

const int N = 3e5 + 9;

int T, low[N], dis[N], art[N];
vector<int> g[N];
void dfs(int u, int pre = 0) {
    low[u] = dis[u] = ++T;
    int child = 0;
    for(auto v: g[u]) {
        if(!dis[v]) {
            dfs(v, u);
            low[u] = min(low[u], low[v]);
            if(low[v] >= dis[u] && pre != 0) art[u] = 1;
            ++child;
        }
        else if(v != pre) low[u] = min(low[u], dis[v]);
    }
    if(pre == 0 && child > 1) art[u] = 1;
}
int32_t main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    while(1){
        int n, m; cin >> n >> m;
        if(!n) break;
        while(m--) {
            int u, v; cin >> u >> v;
            g[u].push_back(v);
            g[v].push_back(u);
        }
        dfs(1);
        int ans = 0;
        for(int i = 1; i <= n; i++) ans += art[i];
        cout << ans << '\n';
        T = 0; for(int i = 1; i <= n; i++) low[i] = dis[i] = art[i] = 0,
           g[i].clear();
    }
    return 0;
}
```

5 Matematica

5.1 Crivo de Eratostenes

```
// Computa numeros primos entre [2, n] em O(n)
//
// Crivo linear computando spf (smallest prime factor) pra cada numero
// x entre [2, n] e phi(x) (funcao totiente)
// Complexidade: O(n)

int spf[maxn], phi[maxn];
vector<int> primes;
void sieve(int n) {
   phi[1] = 1;
   for (int i = 2; i <= n; i++) {
       if (spf[i] == 0) {
           spf[i] = i;
           primes.emplace_back(i);
           phi[i] = i - 1;
       }
       for (int j = 0; j < (int)primes.size() && i * primes[j] <= n &&
          primes[j] <= spf[i]; j++) {
           spf[i * primes[j]] = primes[j];
           if (primes[j] < spf[i]) phi[i * primes[j]] = phi[i] *
              phi[primes[j]];
           else phi[i * primes[j]] = phi[i] * primes[j];
       }
   }
}
```

5.2 Fast Fourier Transform

```
// Fonte: https://github.com/ShahjalalShohag/code-library
//
// Faz convolucao de dois polinomios
// Complexidade: O(n log(n))
//
// Testado e sem erro de precisao para MAXN = 3e5 e A_i = 1e9

const int N = 3e5 + 9;

const double PI = acos(-1);
struct base {
    double a, b;
    base(double a = 0, double b = 0) : a(a), b(b) {}
    const base operator + (const base &c) const
    { return base(a + c.a, b + c.b); }
    const base operator - (const base &c) const
    { return base(a - c.a, b - c.b); }
    const base operator * (const base &c) const
    { return base(a * c.a - b * c.b, a * c.b + b * c.a); }
};
void fft(vector<base> &p, bool inv = 0) {
    int n = p.size(), i = 0;
    for(int j = 1; j < n - 1; ++j) {
        for(int k = n >> 1; k > (i ^= k); k >>= 1);
        if(j < i) swap(p[i], p[j]);
    }
    for(int l = 1, m; (m = l << 1) <= n; l <<= 1) {
```

```
        double ang = 2 * PI / m;
        base wn = base(cos(ang), (inv ? 1. : -1.) * sin(ang)), w;
        for(int i = 0, j, k; i < n; i += m) {
            for(w = base(1, 0), j = i, k = i + l; j < k; ++j, w = w * wn) {
                base t = w * p[j + l];
                p[j + l] = p[j] - t;
                p[j] = p[j] + t;
            }
        }
    }
    if(inv) for(int i = 0; i < n; ++i) p[i].a /= n, p[i].b /= n;
}
vector<long long> multiply(vector<int> &a, vector<int> &b) {
    int n = a.size(), m = b.size(), t = n + m - 1, sz = 1;
    while(sz < t) sz <<= 1;
    vector<base> x(sz), y(sz), z(sz);
    for(int i = 0 ; i < sz; ++i) {
        x[i] = i < (int)a.size() ? base(a[i], 0) : base(0, 0);
        y[i] = i < (int)b.size() ? base(b[i], 0) : base(0, 0);
    }
    fft(x), fft(y);
    for(int i = 0; i < sz; ++i) z[i] = x[i] * y[i];
    fft(z, 1);
    vector<long long> ret(sz);
    for(int i = 0; i < sz; ++i) ret[i] = (long long) round(z[i].a);
    while((int)ret.size() > 1 && ret.back() == 0) ret.pop_back();
    return ret;
}

long long ans[N];
int32_t main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int n, x; cin >> n >> x;
    vector<int> a(n + 1, 0), b(n + 1, 0), c(n + 1, 0);
    int nw = 0;
    a[0]++; b[n]++;
    long long z = 0;
    for (int i = 1; i <= n; i++) {
        int k; cin >> k;
        nw += k < x;
        a[nw]++; b[-nw + n]++;
        z += c[nw] + !nw; c[nw]++;
    }
    auto res = multiply(a, b);
    for (int i = n + 1; i < res.size(); i++) {
        ans[i - n] += res[i];
    }
    ans[0] = z;
    for (int i = 0; i <= n; i++) cout << ans[i] << ' ';
    cout << '\n';
    return 0;
}
```

5.3 Pollard Rho

```
// Fonte: https://github.com/shahjalalshohag/code-library
//
// Fatora numeros ate 8*10^18
```

```cpp
// Complexidade: O(n ^ (1/4))

using ll = long long;
namespace PollardRho {
    mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
    const int P = 1e6 + 9;
    ll seq[P];
    int primes[P], spf[P];
    inline ll add_mod(ll x, ll y, ll m) {
        return (x += y) < m ? x : x - m;
    }
    inline ll mul_mod(ll x, ll y, ll m) {
        ll res = __int128(x) * y % m;
        return res;
        // ll res = x * y - (ll)((long double)x * y / m + 0.5) * m;
        // return res < 0 ? res + m : res;
    }
    inline ll pow_mod(ll x, ll n, ll m) {
        ll res = 1 % m;
        for (; n; n >>= 1) {
            if (n & 1) res = mul_mod(res, x, m);
            x = mul_mod(x, x, m);
        }
        return res;
    }
    // O(it * (logn)^3), it = number of rounds performed
    inline bool miller_rabin(ll n) {
        if (n <= 2 || (n & 1 ^ 1)) return (n == 2);
        if (n < P) return spf[n] == n;
        ll c, d, s = 0, r = n - 1;
        for (; !(r & 1); r >>= 1, s++) {}
        // each iteration is a round
        for (int i = 0; primes[i] < n && primes[i] < 32; i++) {
            c = pow_mod(primes[i], r, n);
            for (int j = 0; j < s; j++) {
                d = mul_mod(c, c, n);
                if (d == 1 && c != 1 && c != (n - 1)) return false;
                c = d;
            }
            if (c != 1) return false;
        }
        return true;
    }
    void init() {
        int cnt = 0;
        for (int i = 2; i < P; i++) {
            if (!spf[i]) primes[cnt++] = spf[i] = i;
            for (int j = 0, k; (k = i * primes[j]) < P; j++) {
                spf[k] = primes[j];
                if (spf[i] == spf[k]) break;
            }
        }
    }
    // returns O(n^(1/4))
    ll pollard_rho(ll n) {
        while (1) {
            ll x = rnd() % n, y = x, c = rnd() % n, u = 1, v, t = 0;
            ll *px = seq, *py = seq;
            while (1) {
                *py++ = y = add_mod(mul_mod(y, y, n), c, n);
                *py++ = y = add_mod(mul_mod(y, y, n), c, n);
```

```
                if ((x = *px++) == y) break;
                v = u;
                u = mul_mod(u, abs(y - x), n);
                if (!u) return __gcd(v, n);
                if (++t == 32) {
                    t = 0;
                    if ((u = __gcd(u, n)) > 1 && u < n) return u;
                }
            }
            if (t && (u = __gcd(u, n)) > 1 && u < n) return u;
        }
    }
    vector<ll> factorize(ll n) {
        if (n == 1) return vector <ll>();
        if (miller_rabin(n)) return vector<ll> {n};
        vector <ll> v, w;
        while (n > 1 && n < P) {
            v.push_back(spf[n]);
            n /= spf[n];
        }
        if (n >= P) {
            ll x = pollard_rho(n);
            v = factorize(x);
            w = factorize(n / x);
            v.insert(v.end(), w.begin(), w.end());
        }
        return v;
    }
}
int32_t main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    PollardRho::init();
    int t; cin >> t;
    while (t--) {
        ll n; cin >> n;
        auto f = PollardRho::factorize(n);
        sort(f.begin(), f.end());
        cout << f.size() << ' ';
        for (auto x: f) cout << x << ' '; cout << '\n';
    }
    return 0;
}
```

6 Extra

6.1 Config do Vim

```
// .vimrc

 set nu
 set ai
 set ts=4
 set sw=4
 filetype plugin indent on
 inoremap {} {}<Left><Return><Up><End><Return>

 set nohls
 set belloff=all
 syntax on
 set expandtab
 set noshiftround
 set showmode
 set showcmd
```

6.2 Custom Hash

```
// Hash personalizado pra evitar colisao no unordered_map
// Uso: map<int, int, custom_hash> mapa;

 struct custom_hash {
     static uint64_t splitmix64(uint64_t x) {
         x += 0x9e3779b97f4a7c15;
         x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
         x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
         return x ^ (x >> 31);
     }

     size_t operator()(uint64_t x) const {
         static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().
             time_since_epoch().count();
         return splitmix64(x + FIXED_RANDOM);
     }
 };
```

6.3 Gerador aleatorio de inteiros em [l, r]

```
 mt19937 rng(chrono::steady_clock::now() .time_since_epoch().count());

 ll uniform(ll l, ll r){
    uniform_int_distribution<int> uid(l, r);
    return uid(rng);
 }
```

6.4 Mint

```
// Inteiro automaticamente modulado

 const int mod = 998244353;
```

```
struct Mint {
    int val;
    Mint(int v = 0) { val = v % mod; }
    bool operator == (Mint o) { return val == o.val; }
    int operator * (Mint o) { return (((ll)val * o.val) % mod); }
    int operator + (Mint o) { return ((ll)val + o.val) % mod; }
    int operator - (Mint o) { return ((ll)val - o.val + mod) % mod; }
    int operator ^ (ll o) { return pwr(val, o); }
    int pwr(Mint b, ll e) {
        Mint res; for (res = 1; e; e >>= 1, b = b * b) if (e & 1) res = res *
            res;
        return res.val;
    }
};
```

6.5 Rand C++

```
mt19937 rng(chrono::steady_clock::now() .time_since_epoch().count());
```

6.6 Script de stress test

```
set -e
g++ -O2 code.cpp -o code
g++ -O2 brute.cpp -o brute
g++ -O2 gen.cpp -o gen

for((i = 1; ; ++i)); do
    ./gen > input_file
    ./code < input_file > myAnswer
    ./brute < input_file > correctAnswer
    diff myAnswer correctAnswer > /dev/null || break
    echo "Passed test: "  $i
done

echo "WA on the following test:"
cat input_file
echo "Your answer is:"
cat myAnswer
echo "Correct answer is:"
cat correctAnswer
```

6.7 Script pra rodar C++

```
// chmod +x run
// ./run A.cpp

#!/bin/bash
g++ --std=c++20 -Wall -O2 -fsanitize=address,undefined $1 && ./a.out
```

6.8 Template C++

```
#include <bits/stdc++.h>
#define endl '\n'

using namespace std;
```

```
typedef long long ll;

void solve(){

}

signed main(){
   ios_base::sync_with_stdio(0); cin.tie(0);
   solve();
}
```

6.9 Template de debug simples

```
void _print() { }
template<typename T, typename... U> void _print(T a, U... b) {
    if(sizeof... (b)){
        cerr << a << ", ";
        _print(b...);
    } else cerr << a;
}
#define debug(x...) cerr << "[" << #x << "] = [", _print(x), cerr << "]" <<
   endl

// #define debug(...)
```