

Team Notebook

bashem

May 23, 2023

Contents

1 Concepts	2	4 General	5	8 QueryUpdate	11
1.1 General	2	4.1 debug	5	8.1 BIT	11
1.1.1 customtype	2	4.2 gnuDS	6	8.2 LazySegTree	12
1.1.2 recursiveLambda	2	4.3 GreatestTemplate	6	8.3 MosAlgo	13
1.2 Graph	3	5 Hashing	6	8.4 SegTree	14
1.2.1 edgeRemoveCC	3	5.1 Hashing	6	8.5 SparseTable	15
1.2.2 treerooting	4	5.2 UnorderedMap	8	9 String	16
2 DataStructure	4	6 hello_{test}	8	9.1 <i>zfunction</i>	16
2.1 LinceContainer	4	7 NumberTheory	8	10 test	16
3 Functional	5	7.1 Combinatrics	8		
3.1 Graph	5	3.1.1 LCA	5	7.2 ModInt	9
				7.3 PrimePhiSieve	10
				7.4 PrimeSieve	11

1 Concepts

1.1 General

1.1.1 customtype

```
// credit : https://codeforces.com/blog/entry/96040?#comment-851247
#include <iostream>
#include <compare>

struct Fraction {
    int p, q;

    Fraction (int _p, int _q) : p(_p), q(_q) {}

    std::strong_ordering operator<=> (const Fraction &oth)
        const {
            return p * oth.q <=> q * oth.p;
        }
};

int main () {
    Fraction u (1, 3);
    Fraction v (2, 5);

    // all these are automatically generated by the compiler!
    std::cout << (u < v) << std::endl;
    std::cout << (u > v) << std::endl;
    std::cout << (u <= v) << std::endl;
    std::cout << (u >= v) << std::endl;
}
```

1.1.2 recursiveLambda

```
//https://codeforces.com/blog/entry/89790?#comment-781743
// Solution Link: https://codeforces.com/contest/1344/submission/132034808

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
```

```
#define var(...) " [" << #__VA_ARGS__ ": " << (__VA_ARGS__) << "]"
#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define sz(x) ((int)x.size())
#define endl "\n"

void runCase([[maybe_unused]] const int &TC)
{
    int n, m;
    cin >> n >> m;

    vector<vector<char>> g(n, vector<char>(m));
    vector<vector<int>> vis(n, vector<int>(m, 0));

    vector<int> idx = {-1, 1, 0, 0}, idy = {0, 0, -1, 1};

    set<int> rows, cols;
    set<int> rowsF, colsF;

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cin >> g[i][j];
        }
    }

    auto dfs = [&](auto dfs, int x, int y) -> void
    {
        rows.insert(x);
        cols.insert(y);
        rowsF.insert(x);
        colsF.insert(y);

        vis[x][y] = 1;
        for (int i = 0; i < 4; i++)
        {
            int nx = x + idx[i], ny = y + idy[i];

            if (nx >= 0 && ny >= 0 && nx < n && ny < m)
            {
                if (vis[nx][ny] == 0 && g[nx][ny] == '#')
                {
                    dfs(dfs, nx, ny);
                }
            }
        }
    };

    vis[x][y] = 1;
    for (int i = 0; i < 4; i++)
    {
        int nx = x + idx[i], ny = y + idy[i];

        if (nx >= 0 && ny >= 0 && nx < n && ny < m)
        {
            if (vis[nx][ny] == 0 && g[nx][ny] == '#')
            {
                dfs(dfs, nx, ny);
            }
        }
    }
};
```

```
vector<vector<int>> a(n);
vector<vector<int>> b(m);

int cnt = 0;

for (int i = 0; i < n; i++)
{
    for (int j = 0; j < m; j++)
    {
        if (g[i][j] == '#')
        {
            if (vis[i][j] == 0)
            {
                dfs(dfs, i, j);
                cnt++;
            }
            a[i].push_back(j);
            b[j].push_back(i);
        }
    }
}

auto check = [&](vector<int> &vec) -> bool
{
    sort(all(vec));
    for (int j = 1; j < sz(vec); j++)
    {
        if (vec[j] - vec[j - 1] > 1)
        {
            return false;
        }
    }
    return true;
};

for (int i = 0; i < n; i++)
{
    if (!check(a[i]))
    {
        cout << -1 << endl;
        return;
    }
}

for (int i = 0; i < m; i++)
{
    if (!check(b[i]))
    {
        cout << -1 << endl;
    }
}
```

```

    return;
}
}

for (int i = 0; i < n; i++)
{
    for (int j = 0; j < m; j++)
    {
        if (rowsF.find(i) == rowsF.end() && colsF.find(j) ==
            colsF.end())
        {
            rows.insert(i);
            cols.insert(j);
        }
    }
}

if (sz(rows) == n && sz(cols) == m)
{
    cout << cnt << endl;
}
else
{
    cout << -1 << endl;
}
}

int main()
{
    ios_base::sync_with_stdio(false), cin.tie(0);

    int t = 1;
    // cin >> t;

    for (int tc = 1; tc <= t; tc++)
        runCase(tc);

    return 0;
}

```

1.2 Graph

1.2.1 edgeRemoveCC

// Problem: <https://codeforces.com/contest/1242/problem/B>

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```

typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

#define var(...) " [" << #__VA_ARGS__ ": " << (__VA_ARGS__)
    << "]"
#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define sz(x) ((int)x.size())
#define vec vector
#define endl "\n"

class DSU
{
    std::vector<int> p, csz;

public:
    DSU() {}

    DSU(int dsz) // Max size
    {
        //Default empty
        p.resize(dsz + 5, 0), csz.resize(dsz + 5, 0);

        init(dsz);
    }

    void init(int n)
    {
        // n = size
        for (int i = 0; i <= n; i++)
        {
            p[i] = i, csz[i] = 1;
        }
    }

    //Return parent Recursively
    int get(int x)
    {
        if (p[x] != x)
            p[x] = get(p[x]);

        return p[x];
    }

    // Return Size
    int getSize(int x) { return csz[get(x)]; }
    // Return if Union created Successfully or false if they
    // are already in Union

```

```

bool merge(int x, int y)
{
    x = get(x), y = get(y);
    if (x == y)
        return false;

    if (csz[x] > csz[y])
        std::swap(x, y);

    p[x] = y;
    csz[y] += csz[x];

    return true;
}
};

void runCase([[maybe_unused]] const int &TC)
{
    int n, m;
    cin >> n >> m;

    auto g = vec(n + 1, set<int>());

    auto dsu = DSU(n + 1);

    for (int i = 0; i < m; i++)
    {
        int u, v;
        cin >> u >> v;

        g[u].insert(v);
        g[v].insert(u);
    }

    set<int> elligible;

    for (int i = 1; i <= n; i++)
    {
        elligible.insert(i);
    }

    int i = 1;
    int cnt = 0;

    while (sz(elligible))
    {
        cnt++;
        queue<int> q;
        q.push(*elligible.begin());
        elligible.erase(elligible.begin());
    }
}

```

```

while (sz(q))
{
    int fr = q.front();
    q.pop();

    auto v = eligible.begin();

    while (v != eligible.end())
    {
        if (g[fr].find(*v) == g[fr].end())
        {
            q.push(*v);
            v = eligible.erase(v);
        }
        else
        {
            v++;
        }
    }
}

cout << cnt - 1 << endl;
}

int main()
{
    ios_base::sync_with_stdio(false), cin.tie(0);

    int t = 1;
    //cin >> t;

    for (int tc = 1; tc <= t; tc++)
        runCase(tc);

    return 0;
}

```

1.2.2 treerooting

```

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

```

```

#define faster ios_base::sync_with_stdio(false), cin.tie(0)
#define read freopen("in.txt", "r", stdin)
#define write freopen("out.txt", "w", stdout)
#define var(...) " [" << #__VA_ARGS__ " : " << (__VA_ARGS__) << "]"
#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define endl "\n"

const int N = 2e5 + 5;

vector<int> g[N];
ll sz[N], dist[N], sum[N];

void dfs(int s, int p)
{
    sz[s] = 1;
    dist[s] = 0;
    for (int nxt : g[s])
    {
        if (nxt == p)
            continue;
        dfs(nxt, s);
        sz[s] += sz[nxt];
        dist[s] += (dist[nxt] + sz[nxt]);
    }
}

void dfs1(int s, int p)
{
    if (p != 0)
    {
        ll my_size = sz[s];
        ll my_contrib = (dist[s] + sz[s]);

        sum[s] = sum[p] - my_contrib + sz[1] - sz[s] + dist[s];
    }
    for (int nxt : g[s])
    {
        if (nxt == p)
            continue;
        dfs1(nxt, s);
    }
}

// problem link: https://cses.fi/problemset/task/1133

int main()
{

```

```

    faster;

    int n;
    cin >> n;

    for (int i = 1, u, v; i < n; i++)
        cin >> u >> v, g[u].push_back(v), g[v].push_back(u);

    dfs(1, 0);

    sum[1] = dist[1];

    dfs1(1, 0);

    for (int i = 1; i <= n; i++)
        cout << sum[i] << " ";
    cout << endl;

    return 0;
}

```

2 DataStructure

2.1 LinceContainer

```

/*
source: https://github.com/kth-competitive-programming/
kactl/blob/master/content/data-structures/LineContainer
.h
returns max y
*/
struct Line {
    mutable ll k, m, p;
    bool operator<(const Line& o) const { return k < o.k; }
    bool operator<(ll x) const { return p < x; }
};

struct LineContainer : multiset<Line, less<>> {
    // (for doubles, use inf = 1/.0, div(a,b) = a/b)
    static const ll inf = LLONG_MAX;
    ll div(ll a, ll b) { // floored division
        return a / b - ((a ^ b) < 0 && a % b); }
    bool isect(iterator x, iterator y) {
        if (y == end()) return x->p = inf, 0;
        if (x->k == y->k) x->p = x->m > y->m ? inf : -inf;
        else x->p = div(y->m - x->m, x->k - y->k);
        return x->p >= y->p;
    }
}

```

```

void add(ll k, ll m) {
    auto z = insert({k, m, 0}), y = z++, x = y;
    while (isect(y, z)) z = erase(z);
    if (x != begin() && isect(--x, y)) isect(x, y = erase(y));
    while ((y = x) != begin() && (--x)->p >= y->p)
        isect(x, erase(y));
}

ll query(ll x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return l.k * x + l.m;
}
};

```

3 Functional

3.1 Graph

3.1.1 LCA

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```

#define faster ios_base::sync_with_stdio(false), cin.tie(0),
    cout.tie(0)
#define read freopen("in.txt", "r", stdin)
#define write freopen("out.txt", "w", stdout)
#define var(...) " [" << #__VA_ARGS__ ": " << (__VA_ARGS__)
    << "]"
#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define endl "\n"

```

```

// For more: https://cp-algorithms.com/graph/lca.html
const int MAX_N = 1e5 + 5,
    LOG = 20;

```

```

int up[MAX_N][LOG];
int depth[MAX_N], euler[MAX_N * 2], timer = 0;

```

```
vector<int> g[MAX_N];
```

```

void dfs(int curr, int p)
{
    euler[++timer] = curr;
    for (int next : g[curr])
    {

```

```

        if (next == p)
            continue;
        depth[next] = depth[curr] + 1;
        up[next][0] = curr;
        for (int j = 1; j < LOG; j++)
            up[next][j] = up[up[next][j - 1]][j - 1];
        dfs(next, curr);
        euler[++timer] = curr;
    }
}

```

```

int getLCA(int a, int b)
{
    if (depth[a] < depth[b])
        swap(a, b);

    int k = depth[a] - depth[b];
    for (int j = LOG - 1; j >= 0; j--)
    {
        if (k & (1 << j))
            a = up[a][j];
    }
}

```

```

if (a == b)
    return a;

for (int j = LOG - 1; j >= 0; j--)
    if (up[a][j] != up[b][j])
    {
        a = up[a][j];
        b = up[b][j];
    }

return up[a][0];
}

```

```

int getDist(int a, int b)
{
    return depth[a] + depth[b] - 2 * depth[getLCA(a, b)];
}

```

```

int main()
{
    faster;

    int t;
    cin >> t;

    while (t--)
    {

```

```

    }

    return 0;
}

```

4 General

4.1 debug

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```

template <typename T, typename C = typename T::value_type>
typename enable_if<!is_same<T, string>::value, ostream &::
    type operator<<(ostream &out, const T &c)
{
    for (auto it = c.begin(); it != c.end(); it++)
        out << (it == c.begin() ? "{ " : ", ") << *it;
    return out << (c.empty() ? "{ " : ", ") << "}";
}

```

```

template <typename T, typename S>
ostream &operator<<(ostream &out, const pair<T, S> &p)
{
    return out << "{" << p.first << ", " << p.second << "}";
}

```

```

#define dbg(...) _dbg_print(#__VA_ARGS__, __VA_ARGS__)
#define dbg_line cout << "\n#####\n";

```

```

template <typename Arg1>
void _dbg_print(const char *name, Arg1 &&arg1)
{
    if (name[0] == ' ')
        name++;
    cout << "[" << name << ": " << arg1 << "]"
        << "\n";
}

```

```

template <typename Arg1, typename... Args>
void _dbg_print(const char *names, Arg1 &&arg1, Args &&...
    args)
{
    const char *comma = strchr(names + 1, ',');
    cout << "[";
    cout.write(names, comma - names) << ": " << arg1 << "] ";
    _dbg_print(comma + 1, args...);
}

```

}

4.2 gnuDS

```
#include <bits/stdc++.h>

#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define endl "\n"

#include <ext/pb_ds/assoc_container.hpp> // Common file

// using namespace __gnu_pbds;

// https://codeforces.com/blog/entry/11080
// cout<<*X.find_by_order(4)<<endl; // 16
// cout<<(end(X)==X.find_by_order(6))<<endl; // true
// cout<<X.order_of_key(-5)<<endl; // 0
template <typename T, typename order = std::less<T>>
using ordered_set = __gnu_pbds::tree<T, __gnu_pbds::
    null_type, order, __gnu_pbds::rb_tree_tag, __gnu_pbds::
    tree_order_statistics_node_update>;

int main()
{
    ordered_set<int> X;

    std::cout << *X.find_by_order(4) << endl; // 16
    std::cout << (std::end(X) == X.find_by_order(6)) << endl;
    // true
    std::cout << X.order_of_key(-5) << endl; // 0

    return 0;
}
```

4.3 GreatestTemplate

```
#if defined LOCAL && !defined ONLINE_JUDGE
#include "debug.cpp"
#else
#include <bits/stdc++.h>
using namespace std;
#define dbg(...) ;
#endif

typedef long long ll;
typedef pair<int, int> pii;
```

```
typedef pair<ll, ll> pll;

#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define len(x) ((int)(x).size())
#define vec vector

inline bool read(auto &...a) { return (((cin >> a) ? true :
    false) && ...); }

inline void print(const auto &...a) { ((cout << a), ...); }
inline void println(const auto &...a) { print(a..., '\n'); }

template<typename T>
T set_bit(T n, int bit)
{
    return n | (((T)1) << bit);
}

template<typename T>
T reset_bit(T n, int bit)
{
    return n & ~(((T)1) << bit);
}

template<typename T>
bool get_bit(T n, int bit)
{
    return (n & (((T)1) << bit)) != 0;
}

ll big_mod(ll a, ll p, ll m) {
    ll res = 1 % m, x = a % m;
    while (p > 0)
        res = ((p & 1) ? ((res * x) % m) : res), x = ((x * x) % m)
        , p >>= 1;
    return res;
}

// random number generator
// shuffle => shuffle(all(a), rng);
mt19937 rng(chrono::steady_clock::now().time_since_epoch().
    count());
int ran(int l, int r)
{
    return uniform_int_distribution<int>(l, r)(rng);
}

template <typename T>
using minheap = priority_queue<T, vector<T>, greater<T>>;
```

```
template <typename T, typename F>
pair<T, F> operator+(pair<T, F> a, pair<T, F> b)
{
    return {a.first + b.first, a.second + b.second};
}

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

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

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

void run_case([[maybe_unused]] const int &TC)
{
}

int main()
{
    ios_base::sync_with_stdio(false), cin.tie(0);

    int tt = 1;
    read(tt);

    for (int tc = 1; tc <= tt; tc++)
        run_case(tc);

    return 0;
}
```

5 Hashing

5.1 Hashing

```
#include <bits/stdc++.h>
```

```

using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

#define faster ios_base::sync_with_stdio(false), cin.tie(0), cout.tie(0)
#define read freopen("in.txt", "r", stdin)
#define write freopen("out.txt", "w", stdout)
#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define endl "\n"

const int PRIMES[] = { 2147462393, 2147462419, 2147462587,
    2147462633, 2147462747, 2147463167, 2147463203,
    2147463569, 2147463727, 2147463863, 2147464211,
    2147464549, 2147464751, 2147465153, 2147465563,
    2147465599, 2147465743, 2147465953, 2147466457,
    2147466463, 2147466521, 2147466721, 2147467009,
    2147467057, 2147467067, 2147467261, 2147467379,
    2147467463, 2147467669, 2147467747, 2147468003,
    2147468317, 2147468591, 2147468651, 2147468779,
    2147468801, 2147469017, 2147469041, 2147469173,
    2147469229, 2147469593, 2147469881, 2147469983,
    2147470027, 2147470081, 2147470177, 2147470673,
    2147470823, 2147471057, 2147471327, 2147471581,
    2147472137, 2147472161, 2147472689, 2147472697,
    2147472863, 2147473151, 2147473369, 2147473733,
    2147473891, 2147473963, 2147474279, 2147474921,
    2147474929, 2147475107, 2147475221, 2147475347,
    2147475397, 2147475971, 2147476739, 2147476769,
    2147476789, 2147476927, 2147477063, 2147477107,
    2147477249, 2147477807, 2147477933, 2147478017,
    2147478521 };

// ll base_pow, base_pow_1;
ll base1 = 43, base2 = 47, mod1 = 1e9 + 7, mod2 = 1e9 + 9;

// **** Enable this function for codeforces
void generateRandomBM()
{
    unsigned int seed = (unsigned int)chrono::system_clock::now()
        ().time_since_epoch().count();
    srand(seed); /// to avoid getting hacked in CF, comment
        this line for easier debugging

    int q_len = (sizeof(PRIMES) / sizeof(PRIMES[0])) / 4;
    base1 = PRIMES[rand() % q_len];

```

```

    mod1 = PRIMES[rand() % q_len + q_len];
    base2 = PRIMES[rand() % q_len + 2 * q_len];
    mod2 = PRIMES[rand() % q_len + 3 * q_len];
}

struct Hash
{
public:
    vector<int> base_pow, f_hash, r_hash;
    ll base, mod;

    Hash() {}
    // Update it make it more dynamic like segTree class and DSU
    Hash(int mxSize, ll _base, ll _mod) // Max size
    {
        this->base = _base;
        this->mod = _mod;
        base_pow.resize(mxSize + 2, 1), f_hash.resize(mxSize + 2, 0), r_hash.resize(mxSize + 2, 0);

        for (int i = 1; i <= mxSize; i++)
        {
            base_pow[i] = (int)(base_pow[i - 1] * base % mod);
        }
    }

    void init(string s)
    {
        int n = (int)s.size();

        for (int i = 1; i <= n; i++)
        {
            f_hash[i] = (int)((f_hash[i - 1] * base + int(s[i - 1])) % mod);
        }

        for (int i = n; i >= 1; i--)
        {
            r_hash[i] = (int)((r_hash[i + 1] * base + int(s[i - 1])) % mod);
        }
    }

    int forward_hash(int l, int r)
    {
        int h = (int)(f_hash[r + 1] - (1LL * base_pow[r - 1 + 1] * f_hash[l]) % mod);
        return h < 0 ? (int)mod + h : h;
    }
}

```

```

int reverse_hash(int l, int r)
{
    int h = (int)(r_hash[l + 1] - (1LL * base_pow[r - 1 + 1] * r_hash[r + 2]) % mod);
    return h < 0 ? (int)mod + h : h;
}

};

class DHash
{
public:
    Hash sh1, sh2;
    DHash() {}

    DHash(int mx_size)
    {
        sh1 = Hash(mx_size, base1, mod1);
        sh2 = Hash(mx_size, base2, mod2);
    }

    void init(string s)
    {
        sh1.init(s);
        sh2.init(s);
    }

    ll forward_hash(int l, int r)
    {
        return (ll(sh1.forward_hash(l, r)) << 32) | (sh2.forward_hash(l, r));
    }

    ll reverse_hash(int l, int r)
    {
        return ((ll(sh1.reverse_hash(l, r)) << 32) | (sh2.reverse_hash(l, r)));
    }
};

int main()
{
    faster;
    // For codeforces uncomment generateRandomBM
    // ****//
    //generateRandomBM();
    //****//

    int t;
    cin >> t;
}

```

```

while (t--)
{
}

return 0;
}

```

5.2 UnorderedMap

```

#include <bits/stdc++.h>

// For gp_hash_table
#include <ext/pb_ds/assoc_container.hpp>

using namespace __gnu_pbds;

using namespace std;

typedef long long ll;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;

#define faster ios_base::sync_with_stdio(false), cin.tie(0), cout.tie(0)
#define read freopen("in.txt","r",stdin)
#define write freopen("out.txt","w",stdout)
#define var(...) " [" << #__VA_ARGS__ " ": " << (__VA_ARGS__) << "]"
#define mem(x,n) memset(x,n,sizeof(x))
#define all(x) x.begin(), x.end()
#define endl "\n"

// Reference: https://codeforces.com/blog/entry/62393

struct custom_hash
{
    static uint64_t splitmix64(uint64_t x)
    {
        // http://xorshift.di.unimi.it/splitmix64.c
        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);
}
};

int main()
{
    faster;

    // Example Use
    unordered_map<int, int, custom_hash> mp;

    // Faster
    gp_hash_table<int, int, custom_hash> mp;

    return 0;
}

```

6 hello_{test}

```

#include <gtest/gtest.h>

// Demonstrate some basic assertions.
TEST(HelloTest, BasicAssertions) {
    // Expect two strings not to be equal.
    EXPECT_STRNE("hello", "world");
    // Expect equality.
    EXPECT_EQ(7 * 6, 42);
}

```

7 NumberTheory

7.1 Combinatrics

```

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

#define faster ios_base::sync_with_stdio(false), cin.tie(0)

```

```

#define read freopen("in.txt", "r", stdin)
#define write freopen("out.txt", "w", stdout)
#define var(...) " [" << #__VA_ARGS__ " ": " << (__VA_ARGS__) << "]"
#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define endl "\n"

const int N = 1e6, MOD = 998244353;

struct Combinatrics
{
    vector<ll> fact, fact_inv, inv;
    ll mod, nl;

    Combinatrics() {}

    Combinatrics(ll n, ll mod)
    {
        this->nl = n;
        this->mod = mod;
        fact.resize(n + 1, 1), fact_inv.resize(n + 1, 1), inv.
            resize(n + 1, 1);
        init();
    }

    void init()
    {
        fact[0] = 1;

        for (int i = 1; i <= nl; i++)
        {
            fact[i] = (fact[i - 1] * i) % mod;
        }

        inv[0] = inv[1] = 1;
        for (int i = 2; i <= nl; i++)
            inv[i] = inv[mod % i] * (mod - mod / i) % mod;

        fact_inv[0] = fact_inv[1] = 1;

        for (int i = 2; i <= nl; i++)
            fact_inv[i] = (inv[i] * fact_inv[i - 1]) % mod;
    }

    ll ncr(ll n, ll r)
    {
        if (n < r)
        {
            return 0;
        }
    }
}

```



```

}

if (n > nl)
    return ncr(n, r, mod);
return (((fact[n] * 1LL * fact_inv[r]) % mod) * 1LL *
        fact_inv[n - r]) % mod;
}

ll npr(ll n, ll r)
{
    if (n < r)
    {
        return 0;
    }

    if (n > nl)
        return npr(n, r, mod);
    return (fact[n] * 1LL * fact_inv[n - r]) % mod;
}

ll big_mod(ll a, ll p, ll m = -1)
{
    m = (m == -1 ? mod : m);
    ll res = 1 % m, x = a % m;
    while (p > 0)
        res = ((p & 1) ? ((res * x) % m) : res), x = ((x * x) % m), p >>= 1;
    return res;
}

ll mod_inv(ll a, ll p)
{
    return big_mod(a, p - 2, p);
}

ll ncr(ll n, ll r, ll p)
{
    if (n < r)
        return 0;
    if (r == 0)
        return 1;
    return (((fact[n] * mod_inv(fact[r], p)) % p) * mod_inv(
        fact[n - r], p)) % p;
}

ll npr(ll n, ll r, ll p)
{
    if (n < r)
        return 0;
    if (r == 0)

```

```

        return 1;
    return (fact[n] * mod_inv(fact[n - r], p)) % p;
}

};

int main()
{
    faster;

    int t;
    cin >> t;

    Combinatrics comb(N, MOD);

    while (t--)
    {
    }

    return 0;
}

```

7.2 ModInt

```

#include <bits/stdc++.h>
// Tested By Ac
// submission : https://atcoder.jp/contests/abc238/submissions/29247261
// problem : https://atcoder.jp/contests/abc238/tasks/abc238\_c

template <const int MOD>
struct ModInt
{
    int val;
    ModInt() { val = 0; }
    ModInt(long long v) { v += (v < 0 ? MOD : 0), val = (int)(v % MOD); }
    ModInt &operator+=(const ModInt &rhs)
    {
        val += rhs.val, val -= (val >= MOD ? MOD : 0);
        return *this;
    }
    ModInt &operator-=(const ModInt &rhs)
    {
        val -= rhs.val, val += (val < 0 ? MOD : 0);
        return *this;
    }
    ModInt &operator*=(const ModInt &rhs)
    {

```

```

        val = (int)((val * 1ULL * rhs.val) % MOD);
        return *this;
    }
    ModInt pow(long long n) const
    {
        ModInt x = *this, r = 1;
        while (n)
            r = ((n & 1) ? r * x : r), x = (x * x), n >>= 1;
        return r;
    }
    ModInt inv() const { return this->pow(MOD - 2); }
    ModInt &operator/=(const ModInt &rhs) { return *this = *
        this * rhs.inv(); }
    friend ModInt operator+(const ModInt &lhs, const ModInt &
        rhs) { return ModInt(lhs) += rhs; }
    friend ModInt operator-(const ModInt &lhs, const ModInt &
        rhs) { return ModInt(lhs) -= rhs; }
    friend ModInt operator*(const ModInt &lhs, const ModInt &
        rhs) { return ModInt(lhs) *= rhs; }
    friend ModInt operator/(const ModInt &lhs, const ModInt &
        rhs) { return ModInt(lhs) /= rhs; }
    friend bool operator==(const ModInt &lhs, const ModInt &rhs)
        { return lhs.val == rhs.val; }
    friend bool operator!=(const ModInt &lhs, const ModInt &rhs)
        { return lhs.val != rhs.val; }
    friend std::ostream &operator<<(std::ostream &out, const
        ModInt &m) { return out << m.val; }
    friend std::istream &operator>>(std::istream &in, ModInt &m)
        { return in >> m.val; }
    operator int() const { return val; }
};

const int MOD = 1e9 + 7;
using mint = ModInt<MOD>;

int main()
{
    using namespace std;
    const mint N = 5;

    std::vector<mint> a(5, 0);

    for (mint i = 0; i < N; i += 1)
    {
        std::cin >> a[i];
    }

    cout << (mint)7 - (mint)6 << endl;

    for (mint i = 0; i < N; i += 1)

```

```
{
    std::cout << a[i] << std::endl;
}

return 0;
}
```

7.3 PrimePhiSieve

```
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

#define faster ios_base::sync_with_stdio(false), cin.tie(0), cout.tie(0)
#define read freopen("in.txt", "r", stdin)
#define write freopen("out.txt", "w", stdout)
#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define endl "\n"

struct PrimePhiSieve
{
private:
    ll n;
    vector<ll> primes, phi;
    vector<bool> is_prime;

public:
    PrimePhiSieve() {}

    PrimePhiSieve(ll n)
    {
        this->n = n, is_prime.resize(n + 5, true), phi.resize(n + 5, 1);
        phi_sieve();
    }
    void phi_sieve()
    {
        is_prime[0] = is_prime[1] = false;

        for (ll i = 1; i <= n; i++)
            phi[i] = i;

        for (ll i = 1; i <= n; i++)
```

```
        if (is_prime[i])
        {
            primes.push_back(i);
            phi[i] *= (i - 1), phi[i] /= i;

            for (ll j = i + i; j <= n; j += i)
                is_prime[j] = false, phi[j] /= i, phi[j] *= (i - 1);
        }
    }

    ll get_divisors_count(int number, int divisor)
    {
        return phi[number / divisor];
    }

    vector<pll> factorize(ll num)
    {
        vector<pll> a;
        for (int i = 0; i < (int)primes.size() && primes[i] * 1LL
            * primes[i] <= num; i++)
            if (num % primes[i] == 0)
            {
                int cnt = 0;
                while (num % primes[i] == 0)
                    cnt++, num /= primes[i];
                a.push_back({primes[i], cnt});
            }

        if (num != 1)
            a.push_back({num, 1});
        return a;
    }

    ll get_phi(int n)
    {
        return phi[n];
    }

    // (n/p) * (p-1) => n - (n/p);
    void segmented_phi_sieve(ll l, ll r)
    {
        vector<ll> current_phi(r - l + 1);
        vector<ll> left_over_prime(r - l + 1);

        for (ll i = 1; i <= r; i++)
            current_phi[i - l] = i, left_over_prime[i - l] = i;

        for (ll p : primes)
        {
            ll to = ((l + p - 1) / p) * p;
```

```
            if (to == p)
                to += p;

            for (ll i = to; i <= r; i += p)
            {
                while (left_over_prime[i - l] % p == 0)
                    left_over_prime[i - l] /= p;
                current_phi[i - l] -= current_phi[i - l] / p;
            }
        }

        for (ll i = 1; i <= r; i++)
        {
            if (left_over_prime[i - l] > 1)
                current_phi[i - l] -= current_phi[i - l] /
                    left_over_prime[i - l];
            cout << current_phi[i - l] << endl;
        }
    }

    ll phi_sqrt(ll n)
    {
        ll res = n;

        for (ll i = 1; i * i <= n; i++)
        {
            if (n % i == 0)
            {
                res /= i;
                res *= (i - 1);

                while (n % i == 0)
                    n /= i;
            }
        }

        if (n > 1)
            res /= n, res *= (n - 1);
        return res;
    }

};

int main()
{
    faster;

    int t;
    cin >> t;

    while (t--)
```

```
{
}

return 0;
}
```

7.4 PrimeSieve

```
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

#define faster ios_base::sync_with_stdio(false), cin.tie(0), cout.tie(0)
#define read freopen("in.txt", "r", stdin)
#define write freopen("out.txt", "w", stdout)
#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define endl "\n"

struct PrimeSieve
{
public:
    vector<int> primes;
    vector<bool> isprime;
    int n;

    PrimeSieve() {}

    PrimeSieve(int n)
    {
        this->n = n, isprime.resize(n + 5, true), primes.clear();
        sieve();
    }

    void sieve()
    {
        isprime[0] = isprime[1] = false;

        primes.push_back(2);
        for (int i = 4; i <= n; i += 2)
            isprime[i] = false;

        for (int i = 3; 1LL * i * i <= n; i += 2)
            if (isprime[i])
```

```
                for (int j = i * i; j <= n; j += 2 * i)
                    isprime[j] = false;

        for (int i = 3; i <= n; i += 2)
            if (isprime[i])
                primes.push_back(i);
    }

    vector<pll> factorize(ll num)
    {
        vector<pll> a;
        for (int i = 0; i < (int)primes.size() && primes[i] * 1LL
            * primes[i] <= num; i++)
            if (num % primes[i] == 0)
            {
                int cnt = 0;
                while (num % primes[i] == 0)
                    cnt++, num /= primes[i];
                a.push_back({primes[i], cnt});
            }

        if (num != 1)
            a.push_back({num, 1});
        return a;
    }

    vector<ll> segmented_sieve(ll l, ll r)
    {
        vector<ll> seg_primes;
        vector<bool> current_primes(r - l + 1, true);
        for (ll p : primes)
        {
            ll to = (l / p) * p;
            if (to < l)
                to += p;
            if (to == p)
                to += p;
            for (ll i = to; i <= r; i += p)
            {
                current_primes[i - l] = false;
            }
        }

        for (int i = l; i <= r; i++)
        {
            if (i < 2)
                continue;
            if (current_primes[i - l])
            {
```

```
                seg_primes.push_back(i);
            }
        }
        return seg_primes;
    }
};

int main()
{
    faster;

    int t;
    cin >> t;

    while (t--)
    {
    }

    return 0;
}
```

8 QueryUpdate

8.1 BIT

```
#include <bits/stdc++.h>

#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define endl "\n"

// Insert 0 at the beginning of input array to make it 1
// based indexing
// (ind & -ind) returns least significant bit position k,
// 2^(k-1) (1 based indexing)
// for 101 wilong long return 1; for 1010 wilong long return
// 2;
struct BIT
{
private:
    std::vector<long long> mArray;

public:
    BIT(int sz) // Max size of the array
    {
        mArray.resize(sz + 1, 0);
    }
}
```

```

void build(const std::vector<long long> &list)
{
    for (int i = 1; i <= list.size(); i++)
    {
        mArray[i] = list[i];
    }

    for (int ind = 1; ind <= mArray.size(); ind++)
    {
        int ind2 = ind + (ind & -ind);
        if (ind2 <= mArray.size())
        {
            mArray[ind2] += mArray[ind];
        }
    }

    long long prefix_query(int ind)
    {
        int res = 0;
        for (; ind > 0; ind -= (ind & -ind))
        {
            res += mArray[ind];
        }
        return res;
    }

    long long range_query(int from, int to)
    {
        return prefix_query(to) - prefix_query(from - 1);
    }

    void add(int ind, long long add)
    {
        for (; ind < mArray.size(); ind += (ind & -ind))
        {
            mArray[ind] += add;
        }
    };

    int main()
    {

        return 0;
    }

```

8.2 LazySegTree

```

#include <bits/stdc++.h>

template <typename T, typename F, T (*op)(T, T), F (*
    lazy_to_lazy)(F, F), T (*lazy_to_seg)(T, F, int, int)>
struct LazySegTree
{
private:
    std::vector<T> segt;
    std::vector<F> lazy;
    int n;
    T neutral;
    F lazyE;

    int left(int si) { return si * 2; }
    int right(int si) { return si * 2 + 1; }
    int midpoint(int ss, int se) { return (ss + (se - ss) / 2); }

    T query(int ss, int se, int si, int qs, int qe)
    {
        // **** //
        if (lazy[si] != lazyE)
        {
            T curr = lazy[si];
            lazy[si] = lazyE;
            segt[si] = lazy_to_seg(segt[si], curr, ss, se);

            if (ss != se)
            {
                lazy[left(si)] = lazy_to_lazy(lazy[left(si)], curr);
                lazy[right(si)] = lazy_to_lazy(lazy[right(si)], curr);
            }
        }

        if (se < qs || qe < ss)
            return neutral;

        if (qs <= ss && qe >= se)
            return segt[si];

        int mid = midpoint(ss, se);

        return op(query(ss, mid, left(si), qs, qe), query(mid + 1,
            se, right(si), qs, qe));
    }

    void update(int ss, int se, int si, int qs, int qe, F val)
    {

```

```

        // **** //

        if (lazy[si] != lazyE)
        {
            F curr = lazy[si];
            lazy[si] = lazyE;
            segt[si] = lazy_to_seg(segt[si], curr, ss, se);
            if (ss != se)
            {
                lazy[left(si)] = lazy_to_lazy(lazy[left(si)], curr);
                lazy[right(si)] = lazy_to_lazy(lazy[right(si)], curr);
            }
        }

        if (se < qs || qe < ss)
            return;

        if (qs <= ss && qe >= se)
        {
            // **** //

            segt[si] = lazy_to_seg(segt[si], val, ss, se);

            if (ss != se)
            {
                lazy[left(si)] = lazy_to_lazy(lazy[left(si)], val);
                lazy[right(si)] = lazy_to_lazy(lazy[right(si)], val);
            }
            return;
        }

        int mid = midpoint(ss, se);

        update(mid + 1, se, si * 2 + 1, qs, qe, val);
        update(ss, mid, left(si), qs, qe, val);

        segt[si] = op(segt[left(si)], segt[right(si)]);
    }

public:
    LazySegTree() : n(0) {}

    LazySegTree(int sz, T ini, T _neutral, F _lazyE)
    {
        this->n = sz + 1;
        this->neutral = _neutral;
        this->lazyE = _lazyE;
        segt.resize(n * 4 + 5, ini);
        lazy.resize(n * 4 + 5, _lazyE);
    }

```

```

LazySegTree(const std::vector<T> &arr, T ini, T _neutral, F
    _lazyE) : LazySegTree((int)arr.size(), ini, _neutral,
    _lazyE)
{
    init(arr);
}

void init(const std::vector<T> &arr)
{
    this->n = (int)arr.size();
    for (int i = 0; i < n; i++)
        set(i, i, arr[i]);
}

T get(int qs, int qe)
{
    return query(0, n - 1, 1, qs, qe);
}

void set(int from, int to, F val)
{
    update(0, n - 1, 1, from, to, val);
}
};

int op(int a, int b)
{
    return a + b;
}

int lazy_to_seg(int seg, int lazy_v, int l, int r)
{
    return seg + (lazy_v * (r - l + 1));
}

int lazy_to_lazy(int curr_lazy, int input_lazy)
{
    return curr_lazy + input_lazy;
}

int main()
{
    test("Range Sum",
        [&]() -> bool
        {
            LazySegTree<int, int, op, lazy_to_lazy, lazy_to_seg>
                tree(1e5, 0, 0, 0);

```

```

const int N = 105, M = 1e3;

std::vector<int> a(N, 0);
for (int i = 0; i < N; i++)
{
    a[i] = rng::ran(0, M);
}

tree.init(a);
for (int i = 0; i < 100; i++)
{
    int l = rng::ran(0, N - 1);
    int r = rng::ran(l, N - 1);

    int sum = 0;

    for (int j = l; j <= r; j++)
    {
        sum += a[j];
    }

    if (sum != tree.get(l, r))
    {
        write(l, " ", r, " ", sum, " ", tree.get(l, r), "\n");
        return false;
    }

    int val = rng::ran(-M, M);
    tree.set(l, r, val);

    for (int j = l; j <= r; j++)
    {
        a[j] += val;
    }
}

return true;
});

return 0;
}

```

8.3 MosAlgo

```

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

```

```

typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

#define faster ios_base::sync_with_stdio(false), cin.tie(0)
#define read freopen("in.txt", "r", stdin)
#define write freopen("out.txt", "w", stdout)
#define var(...) " [" << #__VA_ARGS__ ": " << (__VA_ARGS__)
    << "]"
#define mem(x, n) memset(x, n, sizeof(x))
#define all(x) x.begin(), x.end()
#define endl "\n"

const int N = 3e4 + 5;
const int blk = sqrt(N) + 1;

struct Query
{
    int l, r, i;
    bool operator<(const Query q) const
    {
        if (this->l / blk == q.l / blk)
            return this->r < q.r;
        return this->l / blk < q.l / blk;
    }
};

vector<int> mos_algorithm(vector<Query> &queries, vector<
    int> &a)
{
    vector<int> answers(queries.size());
    sort(queries.begin(), queries.end());

    int sza = 1e6 + 5;
    vector<int> freq(sza);

    int cnt = 0;

    auto add = [&](int x) -> void
    {
        freq[x]++;
        if (freq[x] == 1)
            cnt++;
    };

    auto remove = [&](int x) -> void
    {
        freq[x]--;
        if (freq[x] == 0)
            cnt--;
    };

```

```

int l = 0;
int r = -1;
for (Query q : queries)
{
    while (l > q.l)
    {
        l--;
        add(a[l]);
    }
    while (r < q.r)
    {
        r++;
        add(a[r]);
    }
    while (l < q.l)
    {
        remove(a[l]);
        l++;
    }
    while (r > q.r)
    {
        remove(a[r]);
        r--;
    }
    answers[q.i] = cnt;
}

return answers;
}

int main()
{
    faster;

    int n;
    cin >> n;

    vector<int> a(n);
    for (int i = 0; i < n; i++)
        cin >> a[i];

    int q;
    cin >> q;

    vector<Query> qr(q);

    for (int i = 0; i < q; i++)
    {
        int l, r;
        cin >> l >> r;

```

```

        l--, r--;
        qr[i].l = l, qr[i].r = r, qr[i].i = i;
    }

    vector<int> res = mos_algorithm(qr, a);

    for (int i = 0; i < q; i++)
        cout << res[i] << endl;

    return 0;
}

8.4 SegTree

#include <bits/stdc++.h>

#include <gtest/gtest.h>

template <typename T, T (*op)(T, T)>
struct SegTree
{
private:
    std::vector<T> segt;
    int n;
    T e;

    int left(int si) { return si * 2; }
    int right(int si) { return si * 2 + 1; }
    int midpoint(int ss, int se) { return (ss + (se - ss) / 2); }

    T query(int ss, int se, int qs, int qe, int si)
    {
        if (se < qs || qe < ss)
            return e;

        if (qs <= ss && qe >= se)
            return segt[si];

        int mid = midpoint(ss, se);

        return op(query(ss, mid, qs, qe, left(si)), query(mid + 1,
            se, qs, qe, right(si)));
    }

    void update(int ss, int se, int key, int si, T val)
    {
        if (ss == se)

```

```

        {
            segt[si] = val;
            return;
        }

        int mid = midpoint(ss, se);

        if (key > mid)
            update(mid + 1, se, key, right(si), val);
        else
            update(ss, mid, key, left(si), val);

        segt[si] = op(segt[left(si)], segt[right(si)]);
    }

public:
    SegTree() : n(0) {}

    SegTree(int sz, T _e)
    {
        this->e = _e;
        this->n = sz + 1;
        segt.resize(n * 4 + 5, _e);
    }

    SegTree(const std::vector<T> &arr, T _e) : SegTree((int)arr
        .size(), _e)
    {
        init(arr);
    }

    void init(const std::vector<T> &arr)
    {
        this->n = (int)arr.size();
        for (int i = 0; i < n; i++)
            set(i, arr[i]);
    }

    T get(int qs, int qe)
    {
        return query(0, n - 1, qs, qe, 1);
    }

    void set(int key, T val)
    {
        update(0, n - 1, key, 1, val);
    }
};

/*

```

```

1.Class Version of Segment Tree
2.Function Version of Segment Tree
*/

/*
@class Version May need Debuging Never Used Before
*/

// T=>Data Type , e => return if query out of range

/* range minimum

int op(int a, int b)
{
    return min(a, b);
}

SegTree<int, op> minTree(1e5,INT_MAX);
*/

/* range maximum
int op(int a, int b)
{
    return max(a, b);
}

SegTree<int, op> maxTree(1e5,INT_MIN)
*/

/* range sum
int op(int a, int b)
{
    return a + b;
}

SegTree<int, op> sumTree(1e5,0)
*/

/*
@Function version used before
*/

int op(int a, int b)
{
    return std::min(a, b);
}

// random number generator
// shuffle(all(array),rng)

```

```

namespace rng
{
    using namespace std;
    mt19937 rng(chrono::steady_clock::now().time_since_epoch().
        count());
    int ran(int l, int r)
    {
        return uniform_int_distribution<int>(l, r)(rng);
    }
}

TEST(HelloTest, BasicAssertions)
{
    const int N = 110, M = 1e7;

    std::vector<int> a(N);

    for (int i = 0; i < N; i++)
        a[i] = rng::ran(1, M);

    SegTree<int, op> minTree(a, INT_MAX);

    for (int i = 0; i < 100; i++)
    {
        int l = rng::ran(0, N - 1);
        int r = rng::ran(l, N - 1);

        int mnn = a[l];

        for (int j = l; j <= r; j++)
        {
            mnn = std::min(mnn, a[j]);
        }

        EXPECT_EQ(mnn, minTree.get(l, r));

        int v = rng::ran(1, M);
        a[l] = v;
        minTree.set(l, v);
    }
}

```

8.5 SparseTable

```

#include <bits/stdc++.h>

// DRAFT RMQ
template <typename T, T (*op)(T, T)>
struct SparseTable

```

```

{
private:
    std::vector<std::vector<T>> st;
    int n, lg;
    std::vector<int> logs;
    T e;

public:
    SparseTable() : n(0) {}

    SparseTable(int _n)
    {
        this->n = _n;
        int bit = 0;
        while ((1 << bit) <= n)
            bit++;
        this->lg = bit;

        st.resize(n, std::vector<T>(lg));
        logs.resize(n + 1, 0);
        logs[1] = 0;
        for (int i = 2; i <= n; i++)
        {
            logs[i] = logs[i / 2] + 1;
        }
    }

    SparseTable(const std::vector<T> &a) : SparseTable((int)a.
        size())
    {
        init(a);
    }

    void init(const std::vector<T> &a)
    {
        this->n = (int)a.size();

        for (int i = 0; i < n; i++)
        {
            st[i][0] = a[i];
        }

        for (int j = 1; j <= lg; j++)
        {
            for (int i = 0; i + (1 << j) <= n; i++)
            {
                st[i][j] = op(st[i][j - 1], st[std::min(i + (1 << (j - 1)
                    ), n - 1)][j - 1]);
            }
        }
    }
}

```

```

}

T get(int l, int r)
{
    int j = logs[r - l + 1];
    return op(st[l][j], st[r - (1 << j) + 1][j]);
}

int op(int a, int b)
{
    if (a == -1)
        return b;
    if (b == -1)
        return a;
    return std::gcd(a, b);
}

int min(int a, int b)
{
    return std::min(a, b);
}

auto main() -> int
{
    const int N = 1000, M = 1e7;

    std::vector<int> a(N);
    for (int i = 0; i < N; i++)
        a[i] = rng::ran(0, M);

```

```

SparseTable<int, min> SparseTable(a);

test("Range Min RMQ ",
    [&]() -> bool
    {
        for (int i = 0; i < 100; i++)
        {
            int l = rng::ran(0, N - 1);
            int r = rng::ran(l, N - 1);

            int mnn = a[l];

            for (int j = l; j <= r; j++)
            {
                mnn = std::min(mnn, a[j]);
            }

            if (mnn != rmq.get(l, r))
            {
                write(l, " ", r, " ", mnn, " ", rmq.get(l, r), "\n");
                return false;
            }
        }

        return true;
    });

return 0;
}

```

9 String

9.1 *z_function*

```

#include<bits/stdc++.h>

/*
    tested by ac
    submission: https://codeforces.com/contest/432/submission/145953901
    problem: https://codeforces.com/contest/432/problem/D
*/
std::vector<int> z_function(const std::string &s)
{
    int n = (int)s.size();
    std::vector<int> z(n, 0);
    for (int i = 1, l = 0, r = 0; i < n; i++)
    {
        if (i <= r)
            z[i] = std::min(r - i + 1, z[i - l]);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
            z[i]++;
        if (i + z[i] - 1 > r)
            l = i, r = i + z[i] - 1;
    }
    return z;
}

```

10 test