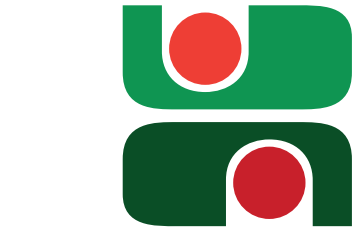


UDESC

Universidade do Estado de Santa Catarina

Não Treinamos o João

Enzo Rodrigues, Eric Grochowicz, João Oliveira

2024-09-11

Contest (1)

.vimrc

7 lines

```
set nu ai si cindent et ts=4 sw=4 so=10 nosm undofile

inoremap {} {}<left><return><up><end><return>
" remap de chaves

au BufReadPost * if line("'\"") > 0 && line("'\"") <= line("$")
      | exe "normal! g'\"" | endif
" volta pro lugar onde estava quando saiu do arquivo
```

run.sh

2 lines

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

hash.sh

3 lines

```
# Hashes a file, ignoring all whitespace and comments. Use for
# verifying that code was correctly typed.
cpp -dD -P -fpreprocessed | tr -d '[:space:]'| md5sum |cut -c-6
```

Mint.cpp

d41d8c, 49 lines

```
template <auto MOD, typename T = decltype(MOD)>
struct Mint {
    using U = long long;
    // se o modulo for long long, usar U = __int128
    using m = Mint<MOD, T>;
    T v;
    Mint(T val = 0) : v(val) {
        assert(sizeof(T) * 2 <= sizeof(U));
        if (v < -MOD || v >= 2 * MOD) v %= MOD;
        if (v < 0) v += MOD;
        if (v >= MOD) v -= MOD;
    }
    Mint(U val) : v(T(val % MOD)) {
        assert(sizeof(T) * 2 <= sizeof(U));
        if (v < 0) v += MOD;
    }
    bool operator==(m o) const { return v == o.v; }
    bool operator<(m o) const { return v < o.v; }
    bool operator!=(m o) const { return v != o.v; }
    m pwr(m b, U e) {
        m res = 1;
        while (e > 0) {
            if (e & 1) res *= b;
            b *= b, e /= 2;
        }
        return res;
    }
    m &operator+=(m o) {
        v -= MOD - o.v;
        if (v < 0) v += MOD;
        return *this;
    }
    m &operator-=(m o) {
        v -= o.v;
        if (v < 0) v += MOD;
        return *this;
    }
    m &operator*=(m o) {
        v = (T)((U)v * o.v % MOD);
        return *this;
    }
    m &operator/=(m o) { return *this *= o.pwr(o, MOD - 2); }
    m &operator^=(U e) { return *this = pwr(*this, e); }
    friend m operator-(m a, m b) { return a -= b; }
    friend m operator+(m a, m b) { return a += b; }
    friend m operator*(m a, m b) { return a *= b; }
    friend m operator/(m a, m b) { return a /= b; }
    friend m operator^(m a, U e) { return a.pwr(a, e); }
};
```

troubleshoot.txt

52 lines

```
Pre-submit:
Write a few simple test cases if sample is not enough.
Are time limits close? If so, generate max cases.
Is the memory usage fine?
Could anything overflow?
Make sure to submit the right file.

Wrong answer:
Print your solution! Print debug output, as well.
Are you clearing all data structures between test cases?
Can your algorithm handle the whole range of input?
Read the full problem statement again.
Do you handle all corner cases correctly?
Have you understood the problem correctly?
Any uninitialized variables?
Any overflows?
Confusing N and M, i and j, etc.?
Are you sure your algorithm works?
What special cases have you not thought of?
Are you sure the STL functions you use work as you think?
Add some assertions, maybe resubmit.
Create some testcases to run your algorithm on.
Go through the algorithm for a simple case.
Go through this list again.
Explain your algorithm to a teammate.
Ask the teammate to look at your code.
Go for a small walk, e.g. to the toilet.
Is your output format correct? (including whitespace)
Rewrite your solution from the start or let a teammate do it.

Runtime error:
Have you tested all corner cases locally?
Any uninitialized variables?
Are you reading or writing outside the range of any vector?
Any assertions that might fail?
Any possible division by 0? (mod 0 for example)
Any possible infinite recursion?
Invalidated pointers or iterators?
Are you using too much memory?
Debug with resubmits (e.g. remapped signals, see Various).

Time limit exceeded:
Do you have any possible infinite loops?
What is the complexity of your algorithm?
Are you copying a lot of unnecessary data? (References)
How big is the input and output? (consider scanf)
Avoid vector, map. (use arrays/unordered_map)
What do your teammates think about your algorithm?

Memory limit exceeded:
What is the max amount of memory your algorithm should need?
Are you clearing all data structures between test cases?
```

Mathematics (2)

2.1 Equations

$$ax^2 + bx + c = 0 \Rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The extremum is given by $x = -b/2a$.

$$\begin{aligned} ax + by &= e \\ cx + dy &= f \end{aligned} \Rightarrow \begin{aligned} x &= \frac{ed - bf}{ad - bc} \\ y &= \frac{af - ec}{ad - bc} \end{aligned}$$

In general, given an equation $Ax = b$, the solution to a variable x_i is given by

$$x_i = \frac{\det A_i'}{\det A}$$

where A_i' is A with the i'th column replaced by b.

2.2 Recurrences

If $a_n = c_1 a_{n-1} + \cdots + c_k a_{n-k}$, and $r_1, \ldots, r_k$ are distinct roots of $x^k - c_1 x^{k-1} - \cdots - c_k$, there are $d_1, \ldots, d_k$ s.t.

$$a_n = d_1 r_1^n + \cdots + d_k r_k^n.$$

Non-distinct roots r become polynomial factors, e.g. $a_n = (d_1 n + d_2) r^n$.

2.3 Trigonometry

$$\sin(v + w) = \sin v \cos w + \cos v \sin w$$
$$\cos(v + w) = \cos v \cos w - \sin v \sin w$$

$$\tan(v + w) = \frac{\tan v + \tan w}{1 - \tan v \tan w}$$
$$\sin v + \sin w = 2 \sin \frac{v + w}{2} \cos \frac{v - w}{2}$$
$$\cos v + \cos w = 2 \cos \frac{v + w}{2} \cos \frac{v - w}{2}$$

$$(V + W) \tan(v - w)/2 = (V - W) \tan(v + w)/2$$

where V, W are lengths of sides opposite angles v, w.

$$a \cos x + b \sin x = r \cos(x - \phi)$$
$$a \sin x + b \cos x = r \sin(x + \phi)$$

where $r = \sqrt{a^2 + b^2}, \phi = \text{atan2}(b, a)$.

2.4 Geometry

2.4.1 Triangles

Side lengths: a, b, c

Semiperimeter: $p = \dfrac{a+b+c}{2}$

Area: $A = \sqrt{p(p-a)(p-b)(p-c)}$

Circumradius: $R = \dfrac{abc}{4A}$

Inradius: $r = \dfrac{A}{p}$

Length of median (divides triangle into two equal-area triangles): $m_a = \frac{1}{2}\sqrt{2b^2 + 2c^2 - a^2}$

Length of bisector (divides angles in two):

$$s_a = \sqrt{bc\left[1 - \left(\frac{a}{b+c}\right)^2\right]}$$

Law of sines: $\dfrac{\sin\alpha}{a} = \dfrac{\sin\beta}{b} = \dfrac{\sin\gamma}{c} = \dfrac{1}{2R}$

Law of cosines: $a^2 = b^2 + c^2 - 2bc\cos\alpha$

Law of tangents: $\dfrac{a+b}{a-b} = \dfrac{\tan\frac{\alpha+\beta}{2}}{\tan\frac{\alpha-\beta}{2}}$

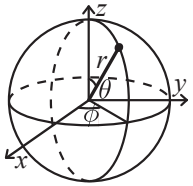
2.4.2 Quadrilaterals

With side lengths a, b, c, d, diagonals e, f, diagonals angle θ, area A and magic flux $F = b^2 + d^2 - a^2 - c^2$:

$$4A = 2ef \cdot \sin\theta = F\tan\theta = \sqrt{4e^2f^2 - F^2}$$

2.4.3 Spherical coordinates

For cyclic quadrilaterals the sum of opposite angles is $180°$, $ef = ac + bd$, and $A = \sqrt{(p-a)(p-b)(p-c)(p-d)}$.



$$\begin{array}{cc} x = r\sin\theta\cos\phi & r = \sqrt{x^2+y^2+z^2} \\ y = r\sin\theta\sin\phi & \theta = \operatorname{acos}(z/\sqrt{x^2+y^2+z^2}) \\ z = r\cos\theta & \phi = \operatorname{atan2}(y,x) \end{array}$$

2.5 Derivatives/Integrals

$$\frac{d}{dx}\arcsin x = \frac{1}{\sqrt{1-x^2}} \qquad \frac{d}{dx}\arccos x = -\frac{1}{\sqrt{1-x^2}}$$

$$\frac{d}{dx}\tan x = 1 + \tan^2 x \qquad \frac{d}{dx}\arctan x = \frac{1}{1+x^2}$$

$$\int \tan ax = -\frac{\ln|\cos ax|}{a} \qquad \int x\sin ax = \frac{\sin ax - ax\cos ax}{a^2}$$

$$\int e^{-x^2} = \frac{\sqrt{\pi}}{2}\operatorname{erf}(x) \qquad \int xe^{ax}dx = \frac{e^{ax}}{a^2}(ax-1)$$

Integration by parts:

$$\int_a^b f(x)g(x)dx = [F(x)g(x)]_a^b - \int_a^b F(x)g'(x)dx$$

2.6 Sums

$$c^a + c^{a+1} + \cdots + c^b = \frac{c^{b+1} - c^a}{c-1}, c \neq 1$$

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$$1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(2n+1)(n+1)}{6}$$

$$1^3 + 2^3 + 3^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$$

$$1^4 + 2^4 + 3^4 + \cdots + n^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

2.7 Series

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots, (-\infty < x < \infty)$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots, (-1 < x \leq 1)$$

$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{2x^3}{32} - \frac{5x^4}{128} + \ldots, (-1 \leq x \leq 1)$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots, (-\infty < x < \infty)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots, (-\infty < x < \infty)$$

2.8 Probability theory

Let X be a discrete random variable with probability $p_X(x)$ of assuming the value x. It will then have an expected value (mean) $\mu = \mathbb{E}(X) = \sum_x xp_X(x)$ and variance $\sigma^2 = V(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2 = \sum_x (x - \mathbb{E}(X))^2 p_X(x)$ where σ is the standard deviation. If X is instead continuous it will have a probability density function $f_X(x)$ and the sums above will instead be integrals with $p_X(x)$ replaced by $f_X(x)$.

Expectation is linear:

$$\mathbb{E}(aX + bY) = a\mathbb{E}(X) + b\mathbb{E}(Y)$$

For independent X and Y,

$$V(aX + bY) = a^2V(X) + b^2V(Y).$$

2.8.1 Discrete distributions

Binomial distribution

The number of successes in n independent yes/no experiments, each which yields success with probability p is $\operatorname{Bin}(n,p)$, $n = 1, 2, \ldots$, $0 \leq p \leq 1$.

$$p(k) = \binom{n}{k}p^k(1-p)^{n-k}$$

$$\mu = np,\ \sigma^2 = np(1-p)$$

$\operatorname{Bin}(n,p)$ is approximately $\operatorname{Po}(np)$ for small p.

First success distribution

The number of trials needed to get the first success in independent yes/no experiments, each which yields success with probability p is $\operatorname{Fs}(p)$, $0 \leq p \leq 1$.

$$p(k) = p(1-p)^{k-1},\ k = 1, 2, \ldots$$

$$\mu = \frac{1}{p},\ \sigma^2 = \frac{1-p}{p^2}$$

Poisson distribution

The number of events occurring in a fixed period of time t if these events occur with a known average rate κ and independently of the time since the last event is $\operatorname{Po}(\lambda)$, $\lambda = t\kappa$.

$$p(k) = e^{-\lambda}\frac{\lambda^k}{k!}, k = 0, 1, 2, \ldots$$

$$\mu = \lambda,\ \sigma^2 = \lambda$$

2.8.2 Continuous distributions

Uniform distribution

If the probability density function is constant between a and b and 0 elsewhere it is $\mathrm{U}(a, b)$, $a < b$.

$$f(x) = \begin{cases} \frac{1}{b-a} & a < x < b \\ 0 & \text{otherwise} \end{cases}$$

$$\mu = \frac{a+b}{2}, \sigma^2 = \frac{(b-a)^2}{12}$$

Exponential distribution

The time between events in a Poisson process is $\mathrm{Exp}(\lambda)$, $\lambda > 0$.

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

$$\mu = \frac{1}{\lambda}, \sigma^2 = \frac{1}{\lambda^2}$$

Normal distribution

Most real random values with mean μ and variance σ^2 are well described by $\mathcal{N}(\mu, \sigma^2)$, $\sigma > 0$.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

If $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ then

$$aX_1 + bX_2 + c \sim \mathcal{N}(\mu_1 + \mu_2 + c, a^2\sigma_1^2 + b^2\sigma_2^2)$$

2.9 Markov chains

A *Markov chain* is a discrete random process with the property that the next state depends only on the current state. Let $X_1, X_2, \ldots$ be a sequence of random variables generated by the Markov process. Then there is a transition matrix $\mathbf{P} = (p_{ij})$, with $p_{ij} = \Pr(X_n = i | X_{n-1} = j)$, and $\mathbf{p}^{(n)} = \mathbf{P}^n \mathbf{p}^{(0)}$ is the probability distribution for X_n (i.e., $p_i^{(n)} = \Pr(X_n = i)$), where $\mathbf{p}^{(0)}$ is the initial distribution.

π is a stationary distribution if $\pi = \pi\mathbf{P}$. If the Markov chain is *irreducible* (it is possible to get to any state from any state), then $\pi_i = \frac{1}{\mathbb{E}(T_i)}$ where $\mathbb{E}(T_i)$ is the expected time between two visits in state i. π_j/π_i is the expected number of visits in state j between two visits in state i.

For a connected, undirected and non-bipartite graph, where the transition probability is uniform among all neighbors, π_i is proportional to node i's degree.

A Markov chain is *ergodic* if the asymptotic distribution is independent of the initial distribution. A finite Markov chain is ergodic iff it is irreducible and *aperiodic* (i.e., the gcd of cycle lengths is 1). $\lim_{k\to\infty} \mathbf{P}^k = \mathbf{1}\pi$.

A Markov chain is an A-chain if the states can be partitioned into two sets $\mathbf{A}$ and $\mathbf{G}$, such that all states in $\mathbf{A}$ are absorbing ($p_{ii} = 1$), and all states in $\mathbf{G}$ leads to an absorbing state in $\mathbf{A}$. The probability for absorption in state $i \in \mathbf{A}$, when the initial state is j, is $a_{ij} = p_{ij} + \sum_{k \in \mathbf{G}} a_{ik} p_{kj}$. The expected time until absorption, when the initial state is i, is $t_i = 1 + \sum_{k \in \mathbf{G}} p_{ki} t_k$.

Data structures (3)

OrderedSet.cpp

Description: Set with operations to find the element by index - find_by_order(i) and to find the index of an element
Usage: `ordered_set<int> seta;`
Time: All operations in $\mathcal{O}(\log N)$.

<ext/pb_ds/assoc_container.hpp>, <ext/pb_ds/tree_policy.hpp> d41d8c, 8 lines

```
using namespace __gnu_pbds;

template <typename T>
using ordered_set =
    tree<T, null_type, less<T>, rb_tree_tag,
         tree_order_statistics_node_update>;

template <typename T, typename U>
using ordered_map = tree<T, U, less<T>, rb_tree_tag,
     tree_order_statistics_node_update>;
```

SegmentTreeBeats.cpp

Description: Updates of max, min and sum in range with queries of sum in range
Usage: `"seg" is already declared globally, just call build`
Time: If sum update is not used: $\mathcal{O}(\log N)$, otherwise $\mathcal{O}(\log^2 N)$

d41d8c, 151 lines

```
const ll INF = 1e18;
struct node {
    ll mi, smi, mx, smx, sum, lazy;
    int fmi, fmx;
    node() {
        mi = smi = INF;
        mx = smx = -INF;
        fmi = 0, fmx = 0, sum = 0, lazy = 0;
    }
    node(ll val) {
        mi = mx = sum = val;
        smi = INF, smx = -INF;
        fmx = fmi = 1;
        lazy = 0;
    }
};

node operator+(node a, node b) {
    node ret;
    ret.sum = a.sum + b.sum;
    if (a.mi == b.mi) {
        ret.mi = a.mi;
        ret.fmi = a.fmi + b.fmi;
        ret.smi = min(a.smi, b.smi);
    } else if (a.mi < b.mi) {
        ret.mi = a.mi;
        ret.fmi = a.fmi;
        ret.smi = min(a.smi, b.mi);
    } else {
        ret.mi = b.mi;
        ret.fmi = b.fmi;
        ret.smi = min(b.smi, a.mi);
    }
    if (a.mx == b.mx) {
        ret.mx = a.mx;
        ret.fmx = a.fmx + b.fmx;
        ret.smx = max(a.smx, b.smx);
    } else if (a.mx > b.mx) {
        ret.mx = a.mx;
        ret.fmx = a.fmx;
        ret.smx = max(b.mx, a.smx);
    } else {
        ret.fmx = b.fmx;
        ret.mx = b.mx;
        ret.smx = max(a.mx, b.smx);
    }
    return ret;
}

struct SegBeats {
    vector<node> t;
    int n;
    void build(int _n) { // pra construir com tamanho, mas
        vazia
        n = _n;
        t.assign(n * 4, node());
    }
    void build(const vector<ll> &v) { // pra construir com
        vector
        n = (int)v.size();
        t.assign(n * 4, node());
        build(1, 0, n - 1, v);
    }
    void build(ll *bg, ll *en) { // pra construir com array de
        C
        build(vector<ll>(bg, en));
    }
    inline int lc(int p) { return 2 * p; }
    inline int rc(int p) { return 2 * p + 1; }
    node build(int p, int l, int r, const vector<ll> &a) {
        if (l == r) return t[p] = node(a[l]);
        int mid = (l + r) >> 1;
        return t[p] = build(lc(p), l, mid, a) + build(rc(p),
            mid + 1, r, a);
    }
    void pushsum(int p, int l, int r, ll x) {
        t[p].sum += (r - l + 1) * x;
        t[p].mi += x;
        t[p].mx += x;
        t[p].lazy += x;
        if (t[p].smi != INF) t[p].smi += x;
        if (t[p].smx != -INF) t[p].smx += x;
    }
    void pushmax(int p, ll x) {
        if (x <= t[p].mi) return;
        t[p].sum += t[p].fmi * (x - t[p].mi);
        if (t[p].mx == t[p].mi) t[p].mx = x;
        if (t[p].smx == t[p].mi) t[p].smx = x;
        t[p].mi = x;
    }
    void pushmin(int p, ll x) {
        if (x >= t[p].mx) return;
        t[p].sum += t[p].fmx * (x - t[p].mx);
        if (t[p].mi == t[p].mx) t[p].mi = x;
        if (t[p].smi == t[p].mx) t[p].smi = x;
        t[p].mx = x;
    }
    void pushdown(int p, int l, int r) {
        if (l == r) return;
        int mid = (l + r) >> 1;
        pushsum(lc(p), l, mid, t[p].lazy);
        pushsum(rc(p), mid + 1, r, t[p].lazy);
```

```
        t[p].lazy = 0;

        pushmax(lc(p), t[p].mi);
        pushmax(rc(p), t[p].mi);

        pushmin(lc(p), t[p].mx);
        pushmin(rc(p), t[p].mx);
    }
    node updatemin(int p, int l, int r, int L, int R, ll x) {
        if (l > R || r < L || x >= t[p].mx) return t[p];
        if (l >= L && r <= R && x > t[p].smx) {
            pushmin(p, x);
            return t[p];
        }
        pushdown(p, l, r);
        int mid = (l + r) >> 1;
        t[p] = updatemin(lc(p), l, mid, L, R, x) + updatemin(rc
            (p), mid + 1, r, L, R, x);
        return t[p];
    }
    node updatemax(int p, int l, int r, int L, int R, ll x) {
        if (l > R || r < L || x <= t[p].mi) return t[p];
        if (l >= L && r <= R && x < t[p].smi) {
            pushmax(p, x);
            return t[p];
        }
        pushdown(p, l, r);
        int mid = (l + r) >> 1;
        t[p] = updatemax(lc(p), l, mid, L, R, x) + updatemax(rc
            (p), mid + 1, r, L, R, x);
        return t[p];
    }
    node updatesum(int p, int l, int r, int L, int R, ll x) {
        if (l > R || r < L) return t[p];
        if (l >= L && r <= R) {
            pushsum(p, l, r, x);
            return t[p];
        }
        pushdown(p, l, r);
        int mid = (l + r) >> 1;
        return t[p] = updatesum(lc(p), l, mid, L, R, x) +
                      updatesum(rc(p), mid + 1, r, L, R, x);
    }
    node query(int p, int l, int r, int L, int R) {
        if (l > R || r < L) return node();
        if (l >= L && r <= R) return t[p];
        pushdown(p, l, r);
        int mid = (l + r) >> 1;
        return query(lc(p), l, mid, L, R) + query(rc(p), mid +
            1, r, L, R);
    }
    ll query(int l, int r) { return query(1, 0, n - 1, l, r).
        sum; }
    void updatemax(int l, int r, ll x) { updatemax(1, 0, n - 1,
         l, r, x); }
    void updatemin(int l, int r, ll x) { updatemin(1, 0, n - 1,
         l, r, x); }
    void updatesum(int l, int r, ll x) { updatesum(1, 0, n - 1,
         l, r, x); }
} seg;
```

SegmentTree2D.cpp

Description: Segment Tree with point update and rectangular range query
Usage: "seg" declared globally
Time: $\mathcal{O}(\log^2 N)$

d41d8c, 47 lines

```
struct SegTree2D {
    ll merge(ll a, ll b) { return a + b; }
    ll neutral = 0;
    int n, m;
    vector<vector<ll>> t;
    void build(int _n, int _m) {
        n = _n, m = _m;
        t.assign(2 * n, vector<ll>(2 * m, neutral));
        for (int i = 2 * n - 1; i >= n; i--)
            for (int j = m - 1; j > 0; j--)
                t[i][j] = merge(t[i][j << 1], t[i][j << 1 | 1])
                    ;
        for (int i = n - 1; i > 0; i--)
            for (int j = 2 * m - 1; j > 0; j--)
                t[i][j] = merge(t[i << 1][j], t[i << 1 | 1][j])
                    ;
    }
    ll inner_query(int idx, int l, int r) {
        ll res = neutral;
        for (l += m, r += m + 1; l < r; l >>= 1, r >>= 1) {
            if (l & 1) res = merge(res, t[idx][l++]);
            if (r & 1) res = merge(res, t[idx][--r]);
        }
        return res;
    }
    // query do ponto (a, b) ate o ponto (c, d), retorna neutro
        se a > c ou b > d
    ll query(int a, int b, int c, int d) {
        ll res = neutral;
        for (a += n, c += n + 1; a < c; a >>= 1, c >>= 1) {
            if (a & 1) res = merge(res, inner_query(a++, b, d))
                ;
            if (c & 1) res = merge(res, inner_query(--c, b, d))
                ;
        }
        return res;
    }
    void inner_update(int idx, int i, ll x) {
        auto &c = t[idx];
        i += m;
        c[i] = x;
        for (i >>= 1; i > 0; i >>= 1) c[i] = merge(c[i << 1], c
            [i << 1 | 1]);
    }
    void update(int i, int j, ll x) {
        i += n;
        inner_update(i, j, x);
        for (i >>= 1; i > 0; i >>= 1) {
            ll val = merge(t[i << 1][j + m], t[i << 1 | 1][j +
                m]);
            inner_update(i, j, val);
        }
    }
} seg;
```

ImplicitTreap.cpp

Description: Does everything
Time: Expected $\mathcal{O}(\log N)$

d41d8c, 124 lines

```
mt19937 rng((uint32_t)chrono::steady_clock::now().
    time_since_epoch().count());
namespace imp_treap {
    using T = ll; // mudar pra int se nao precisar pra melhorar
        a performance
    T merge(T a, T b) { return a + b; }
    T neutral = 0;
    struct node_info {
        node_info *l, *r;
        int y, size;
        T val, acc, add;
        bool rev;
        node_info() { }
        node_info(T _val)
            : l(0), r(0), y(rng()), size(0), val(_val), acc(0),
                add(0), rev(false) { }
    };
    using node = node_info *;
    node root = 0;
    inline int size(node t) { return t ? t->size : 0; }
    inline T acc(node t) { return t ? t->acc : 0; }
    inline bool rev(node t) { return t ? t->rev : false; }
    inline void push(node t) {
        if (!t) return;
        if (rev(t)) {
            t->rev = false;
            swap(t->l, t->r);
            if (t->l) t->l->rev ^= 1;
            if (t->r) t->r->rev ^= 1;
        }
        t->acc += t->add * size(t);
        // t->acc += t->add se for RMQ
        t->val += t->add;
        if (t->l) t->l->add += t->add;
        if (t->r) t->r->add += t->add;
        t->add = 0;
    }
    inline void pull(node t) {
        if (t) {
            push(t->l), push(t->r);
            t->size = size(t->l) + size(t->r) + 1;
            t->acc = merge(t->val, merge(acc(t->l), acc(t->r)))
                ;
        }
    }
    void merge(node &t, node L, node R) {
        push(L), push(R);
        if (!L || !R) {
            t = L ? L : R;
        } else if (L->y > R->y) {
            merge(L->r, L->r, R);
            t = L;
        } else {
            merge(R->l, L, R->l);
            t = R;
        }
        pull(t);
    }
    void split(node t, int pos, node &L, node &R, int add = 0)
        {
        if (!t) {
            L = R = nullptr;
        } else {
            push(t);
            int imp_key = add + size(t->l);
            if (pos <= imp_key) {
                split(t->l, pos, L, t->l, add);
                R = t;
            } else {
                split(t->r, pos, t->r, R, imp_key + 1);
                L = t;
            }
        }
        pull(t);
    }
    inline void insert(node to, int pos) {
        node L, R;
        split(root, pos, L, R);
        merge(L, L, to);
        merge(root, L, R);
    }
    bool remove(node &t, int pos, int add = 0) {
```