BRAC University

BRACU_Crows

Arman Ferdous, Ruhan Habib, Md Mazed Hossain

ICPC World Finals 2024-25

September 4, 2025

Contest (1)

instructions.txt 19 lines

```
1. vi .bashrc: export PATH="$PATH:$HOME/cp"
2. mkdir -p cp/bits/ && cd cp && vi cf (Type)
3. chmod +x cf; restart terminal
4. Type stdc++.h, template.cpp, hash.sh
-------- Kate --------
1. Go to Settings->Configure Kate.
 1. Editing->Default input mode->Vim
 2. Vi input mode->Insert mode->jk = <esc>
 3. Appearance->Turn off dynamic w.w.
 4. Color Themes->Gruvbox
 5. Terminal->Turn off hide console
    (View->Tool Views->Show sidebars is on)
2. Hotkey: Focus Terminal Panel=F4->"Reassign"
-------- Windows --------
1. Using cmd: echo %PATH%. Using Powershell:
     echo $env:PATH
2. Add path using cmd: set PATH=%PATH%;C:\
     Program Files\CodeBlocks\MinGW\bin
   It should be the directory where g++ is.
3. If we're using g++ of CodeBlocks, fsanitize
      won't be available :(
4. Write cf.bat at some directory. Ensure that
      directory is in PATH.
```

cf.sh 3 lines

```
#!/bin/bash
code=$1
g++ ${code}.cpp -o $code -std=c++20 -g -DDeBuG
      -Wall -Wshadow -fsanitize=address,
     undefined && ./$code
```

hash.sh 1 lines

```
cpp -dD -P -fpreprocessed | tr -d '[:space:]'|
      md5sum |cut -c-6
```

stdc++.h 90f4a7, 29 lines

```
#include <bits/stdc++.h>
using namespace std;
#define TT template <typename T

TT,typename=void> struct cerrok:false_type {};
TT> struct cerrok <T, void_t<decltype(cerr <<
     declval<T>() )>> : true_type {};

TT> constexpr void p1 (const T &x);
TT, typename V> void p1(const pair<T, V> &x) {
  cerr << "{"; p1(x.first); cerr << ", ";
  p1(x.second); cerr << "}";
}
TT> constexpr void p1 (const T &x) {
  if constexpr (cerrok<T>::value) cerr << x;
  else { int f = 0; cerr << '{';
    for (auto &i: x)
      cerr << (f++ ? ", " : ""), p1(i);
    cerr << "}";
} }
void p2() { cerr << "]\n"; }
TT, typename... V> void p2(T t, V... v) {
  p1(t);
  if (sizeof...(v)) cerr << ", ";
  p2(v...);
}

#ifdef DeBuG
#define dbg(x...) {cerr << "\t\e[93m"<<
     __func__<<":"<<__LINE__<<" [" << #x << "]
     = ["; p2(x); cerr << "\e[0m";}
#endif
```

template.cpp 640c64, 19 lines

```
// BRACU_Crows
#include "bits/stdc++.h"
using namespace std;

#ifndef DeBuG
  #define dbg(...)
#endif

#define sz(x) (int)(x).size()
#define all(x) begin(x), end(x)
#define rep(i,a,b) for(int i=a;i<(b);++i)
using ll = long long; using pii=pair<int,int>;
using pll = pair<ll,ll>; using vi=vector<int>;
template<class T> using V = vector<T>;

int main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0); cout.tie(0);
}
```

stress.sh 14 lines

```
#!/bin/bash
cf gen > in              # input generator
cf bf < in > exp         # bruteforce
cf code < in > out       # buggy code name

for ((i = 1; ; ++i)) do
   echo $i
   ./gen > in
   ./bf < in > exp
   ./code < in > out     # buggy code name
   diff -w exp out || break
done
# Shows expected first, then user
notify-send "bug found!!!!"
```

cf.bat 5 lines

```
@echo off
setlocal
set prog=%1
g++ %prog%.cpp -o %prog% -DDeBuG -std=c++17 -g
      -Wall -Wshadow && .\%prog%
endlocal
```

Mathematics (2)

2.1 Equations

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

Cramer: Given an equation $Ax = b$, the solution to a variable x_i is given by

$$x_i = \frac{\det A_i'}{\det A} \quad \text{[where } A_i' \text{ is } A \text{ with the } i\text{'th column replaced by } b.]$$

Vieta: Let $P(x) = a_n x^n + ... + a_0$, be a polynomial with complex coefficients and degree n, having complex roots $r_n, ..., r_1$. Then for any integer $0 \le k \le n$,

$$\sum_{1 \le i_1 < i_2 < \ldots < i_k \le n} r_{i_1} r_{i_2} \ldots r_{i_k} = (-1)^k \frac{a_{n-k}}{a_n}$$

Rational Root Theorem: If $\frac{p}{q}$ is a reduced rational root of a polynomial with **integer coeffs**, then $p \mid a_0$ and $q \mid a_n$

2.2 Ceils and Floors

For $x, y \in \mathbb{R}$, $m, n \in \mathbb{Z}$:

- $\lfloor x \rfloor \le x < \lfloor x \rfloor + 1$; $\lceil x \rceil - 1 < x \le \lceil x \rceil$
- $-\lfloor x \rfloor = \lceil -x \rceil$; $-\lceil x \rceil = \lfloor -x \rfloor$
- $\lfloor x + n \rfloor = \lfloor x \rfloor + n$, $\lceil x + n \rceil = \lceil x \rceil + n$
- $\lfloor x \rfloor = m \Leftrightarrow x - 1 < m \le x < m + 1$
- $\lceil x \rceil = n \Leftrightarrow n - 1 < x \le n < x + 1$
- If $n > 0$, $\lfloor \frac{\lfloor x \rfloor + m}{n} \rfloor = \lfloor \frac{x+m}{n} \rfloor$
- If $n > 0$, $\lceil \frac{\lceil x \rceil + m}{n} \rceil = \lceil \frac{x+m}{n} \rceil$
- If $n > 0$, $\lfloor \frac{\lfloor \frac{x}{m} \rfloor}{n} \rfloor = \lfloor \frac{x}{mn} \rfloor$
- If $n > 0$, $\lceil \frac{\lceil \frac{x}{m} \rceil}{n} \rceil = \lceil \frac{x}{mn} \rceil$
- For $m, n > 0$, $\sum_{k=1}^{n-1} \lfloor \frac{km}{n} \rfloor = \frac{(m-1)(n-1) + \gcd(m,n) - 1}{2}$

2.3 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. $a_n = (d_1 n + d_2) r^n$.

2.4 Trigonometry

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

$$(V + W) \tan(\frac{v - w}{2}) = (V - W) \tan(\frac{v + w}{2})$$

V, W are sides opposite to angles v, w.
$a \cos x + b \sin x = r \cos(x - \phi)$
$a \sin x + b \cos x = r \sin(x + \phi)$
where $r = \sqrt{a^2 + b^2}$, $\phi = \operatorname{atan2}(b, a)$.

2.5 Geometry

2.5.1 Triangles

Side lengths: a, b, c

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

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

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

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

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

Law of sines, cosines & tangents:

$\frac{\sin \alpha}{a} = \frac{\sin \beta}{b} = \frac{\sin \gamma}{c} = \frac{1}{2R}$.....(1)

$a^2 = b^2 + c^2 - 2bc \cos \alpha$.....(2)

$\frac{a + b}{a - b} = \frac{\tan((\alpha + \beta)/2)}{\tan((\alpha - \beta)/2)}$.....(3)

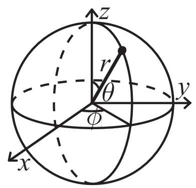
2.5.2 Quadrilaterals

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

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

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

2.5.3 Spherical coordinates



$$x = r\sin\theta\cos\phi \qquad r = \sqrt{x^2+y^2+z^2}$$
$$y = r\sin\theta\sin\phi \qquad \theta = \mathrm{acos}(z/\sqrt{x^2+y^2+z^2})$$
$$z = r\cos\theta \qquad \phi = \mathrm{atan2}(y,x)$$

2.6 Derivatives/Integrals

$$\frac{d}{dx}\arcsin x = \frac{1}{\sqrt{1-x^2}} \qquad \frac{d}{dx}\arccos x = -\frac{1}{\sqrt{1-x^2}}$$
$$\frac{d}{dx}\tan x = 1+\tan^2 x \qquad \frac{d}{dx}\arctan x = \frac{1}{1+x^2}$$
$$\int \tan ax = -\frac{\ln|\cos ax|}{a} \qquad \int xe^{ax}dx = \frac{e^{ax}}{a^2}(ax-1)$$
$$\int e^{-x^2} = \frac{\sqrt{\pi}}{2}\mathrm{erf}(x) \qquad \int x\sin ax = \frac{\sin ax - ax\cos ax}{a^2}$$

Integration by parts:

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

2.7 Sums

$$c^a + c^{a+1} + \cdots + c^b = \frac{c^{b+1}-c^a}{c-1}, c \neq 1$$
$$1+2+3+\cdots+n = \frac{n(n+1)}{2}$$
$$1^2+2^2+3^2+\cdots+n^2 = \frac{n(2n+1)(n+1)}{6} = S_2$$
$$1^3+2^3+3^3+\cdots+n^3 = \frac{n^2(n+1)^2}{4}$$
$$1^4+2^4+3^4+\cdots+n^4 = S_2 \times \frac{3n^2+3n-1}{5}$$

2.8 Series

$$e^x = 1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots, (-\infty < x < \infty)$$
$$\ln(1+x) = x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\ldots, (-1 < x \leq 1)$$
$$\sqrt{1+x} = 1+\frac{x}{2}-\frac{x^2}{8}+\frac{2x^3}{32}-\frac{5x^4}{128}+\ldots, (-1 \leq x \leq 1)$$
$$\sin x = x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\ldots, (-\infty < x < \infty)$$
$$\cos x = 1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots, (-\infty < x < \infty)$$

2.9 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,

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

2.9.1 Discrete distributions

Binomial distribution

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

$$p(k) = \binom{n}{k}p^k(1-p)^{n-k}$$
$$\mu = np, \sigma^2 = np(1-p)$$

$\mathrm{Bin}(n,p)$ is approximately $\mathrm{Po}(np)$ 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 $\mathrm{Fs}(p)$, $0 \leq p \leq 1$.

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

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 $\mathrm{Po}(\lambda)$, $\lambda = t\kappa$.

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

2.9.2 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

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

2.10 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)

OrderStatisticTree.h

Description: A set (not multiset!) with support for finding the n'th element, and finding the index of an element. To get a map, change `null_type`.

Time: $\mathcal{O}(\log N)$

782797, 14 lines

```
#include <bits/extc++.h>
using namespace __gnu_pbds;

template<class T> using Tree=tree<T, null_type
     , less<T>, rb_tree_tag,
     tree_order_statistics_node_update>;

void example() {
  Tree<int> t, t2; t.insert(8);
  auto it = t.insert(10).first;
  assert(it == t.lower_bound(9));
  assert(t.order_of_key(10) == 1);
  assert(t.order_of_key(11) == 2);
  assert(*t.find_by_order(0) == 8);
  t.join(t2); // assuming T < T2 or T > T2,
        merge t2 into t
}
```

HashMap.h

Description: Hash map with mostly the same API as unordered_map, but ~3x faster. Uses 1.5x memory. Initial capacity must be a power of 2 (if provided).

d77092, 7 lines

```
#include <bits/extc++.h>
// To use most bits rather than just the
     lowest ones:
struct chash { // large odd number for C
  const uint64_t C = ll(4e18 * acos(0)) | 71;
  ll operator()(ll x) const { return
        __builtin_bswap64(x*C); }
};
__gnu_pbds::gp_hash_table<ll,int,chash> h({},{
     },{},{},{1<<16});
```

SegmentTree.h

Time: $\mathcal{O}(\log N)$ 9f8f73, 61 lines

```
template<class S> struct segtree {
  int n; V<S> t;
  void init(int _) { n = _; t.assign(n+n-1, S
      ()); }
  void init(const V<S>& v) {
    n = sz(v); t.assign(n + n - 1, S());
    build(0,0,n-1,v);
  } template <typename... T>
  void upd(int l, int r, const T&... v) {
    assert(0 <= l && l <= r && r < n);
    upd(0, 0, n-1, l, r, v...);
  }
  S get(int l, int r) {
    assert(0 <= l && l <= r && r < n);
    return get(0, 0, n-1, l, r);
  }
private:
  inline void push(int u, int b, int e) {
    if (t[u].lazy == 0) return;
    int mid = (b+e)>>1, rc = u+((mid-b+1)<<1);
    t[u+1].upd(b, mid, t[u].lazy);
    t[rc].upd(mid+1, e, t[u].lazy);
    t[u].lazy = 0;
  }
  void build(int u,int b,int e,const V<S>&v) {
    if (b == e) return void(t[u] = v[b]);
    int mid = (b+e)>>1, rc = u+((mid-b+1)<<1);
    build(u+1, b,mid,v); build(rc, mid+1,e,v);
    t[u] = t[u+1] + t[rc];
  } template<typename... T>
  void upd(int u, int b, int e, int l, int r,
      const T&... v) {
    if (l <= b && e <= r) return t[u].upd(b, e
        , v...);
    push(u, b, e);
    int mid = (b+e)>>1, rc = u+((mid-b+1)<<1);
    if (l<=mid) upd(u+1, b, mid, l, r, v...);
    if (mid<r) upd(rc, mid+1, e, l, r, v...);
    t[u] = t[u+1] + t[rc];
  }
  S get(int u, int b, int e, int l, int r) {
    if (l <= b && e <= r) return t[u];
    push(u, b, e);
    S res; int mid = (b+e)>>1, rc = u+((mid-b
        +1)<<1);
    if (r<=mid) res = get(u+1, b, mid, l, r);
    else if (mid<l) res = get(rc,mid+1,e,l,r);
    else res = get(u+1, b, mid, l, r) + get(rc
        , mid+1, e, l, r);
    t[u] = t[u+1] + t[rc]; return res;
  }
}; // Hash upto here = 773c09
/* (1) Declaration:
Create a node class. Now, segtree<node> T;
T.init(10) creates everything as node()
Consider using V<node> leaves to build
(2) upd(l, r, ...v): update range [l, r]
order in ...v must be same as node.upd() fn */
struct node {
  ll sum = 0, lazy = 0;
  node () {} // write full constructor
  node operator+(const node &obj) {
    return {sum + obj.sum, 0};    }
  void upd(int b, int e, ll x) {
    sum += (e - b + 1) * x, lazy += x;
} };
```

UnionFindRollback.h

Description: Disjoint-set data structure with undo. If undo is not needed, skip st, time() and rollback().
Usage: `int t = uf.time(); ...; uf.rollback(t);`
Time: $\mathcal{O}(\log(N))$ de4ad0, 21 lines

```
struct RollbackUF {
  vi e; vector<pii> st;
  RollbackUF(int n) : e(n, -1) {}
  int size(int x) { return -e[find(x)]; }
  int find(int x) { return e[x] < 0 ? x : find
      (e[x]); }
  int time() { return sz(st); }
  void rollback(int t) {
    for (int i = time(); i --> t;)
      e[st[i].first] = st[i].second;
    st.resize(t);
  }
  bool join(int a, int b) {
    a = find(a), b = find(b);
    if (a == b) return false;
    if (e[a] > e[b]) swap(a, b);
    st.push_back({a, e[a]});
    st.push_back({b, e[b]});
    e[a] += e[b]; e[b] = a;
    return true;
  }
};
```

Matrix.h

Description: Basic operations on square matrices.
Usage: `Matrix<int, 3> A;`
`A.d = {{{{1,2,3}}, {{4,5,6}}, {{7,8,9}}}};`
`array<int, 3> vec = {1,2,3};`
`vec = (A^N) * vec;`
6ab5db, 26 lines

```
template<class T, int N> struct Matrix {
  typedef Matrix M;
  array<array<T, N>, N> d{};
  M operator*(const M& m) const {
    M a;
    rep(i,0,N) rep(j,0,N)
      rep(k,0,N) a.d[i][j] += d[i][k]*m.d[k][j
          ];
    return a;
  }
  array<T, N> operator*(const array<T, N>& vec
      ) const {
    array<T, N> ret{};
    rep(i,0,N) rep(j,0,N) ret[i] += d[i][j] *
        vec[j];
    return ret;
  }
  M operator^(ll p) const {
    assert(p >= 0);
    M a, b(*this);
    rep(i,0,N) a.d[i][i] = 1;
    while (p) {
      if (p&1) a = a*b;
      b = b*b;
      p >>= 1;
    }
    return a;
  }
};
```

LineContainer.h

Description: Container where you can add lines of the form kx+m, and query maximum values at points x. Useful for dynamic programming ("convex hull trick").
Time: $\mathcal{O}(\log N)$ 8ec1c7, 30 lines

```
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;
  }
};
```

Lichao.h

Description: Add line segment, query minimum y at some x. Provide list of all query x points to constructor (offline solution). Use `add_segment(line, l, r)` to add a line segment $y = ax + b$ defined by $x \in [l, r)$. Use `query(x)` to get min at x.
Time: Both operations are $\mathcal{O}(\log \text{maxn})$. 566134, 43 lines

```
struct LiChaoTree {
  using Line = pair <ll, ll>;
  const ll linf = numeric_limits<ll>::max();
  int n; vector<ll> xl; vector<Line> dat;
  LiChaoTree(const vector<ll>& _xl):xl(_xl){
    n = 1; while(n < xl.size())n <<= 1;
    xl.resize(n,xl.back());
    dat = vector<Line>(2*n-1, Line(0,linf));
  }
  ll eval(Line f,ll x){return f.first * x + f.
      second;}
  void _add_line(Line f,int k,int l,int r){
    while (l != r) {
      int m = (l + r) / 2;
      ll lx = xl[l],mx = xl[m],rx = xl[r - 1];
      Line &g = dat[k];
      if(eval(f,lx) < eval(g,lx) && eval(f,rx)
          < eval(g,rx)) {
        g = f; return;
      }
      if(eval(f,lx) >= eval(g,lx) && eval(f,rx
          ) >= eval(g,rx))
        return;
      if(eval(f,mx) < eval(g,mx))swap(f,g);
      if(eval(f,lx) < eval(g,lx)) k = k * 2 +
          1, r = m;
      else k = k * 2 + 2, l = m;
    }
  }
  void add_line(Line f){_add_line(f,0,0,n);}
  void add_segment(Line f,ll lx,ll rx){
    int l = lower_bound(xl.begin(), xl.end(),
        lx) - xl.begin();
    int r = lower_bound(xl.begin(), xl.end(),
        rx) - xl.begin();
    int a0 = l, b0 = r, sz = 1; l += n;r += n;
    while(l < r){
      if(r & 1) r--, b0 -= sz, _add_line(f,r -
          1,b0,b0 + sz);
      if(l & 1) _add_line(f,l - 1,a0,a0 + sz),
          l++, a0 += sz;
      l >>= 1, r >>= 1, sz <<= 1;
    }
  }
  ll query(ll x) {
    int i = lower_bound(xl.begin(), xl.end(),x
        ) - xl.begin();
    i += n - 1; ll res = eval(dat[i],x);
    while (i) i = (i - 1) / 2, res = min(res,
        eval(dat[i], x));
    return res;
  }
};
```

Treap.h

Description: A short self-balancing tree. It acts as a sequential container with log-time splits/joins, and is easy to augment with additional data.
Time: $\mathcal{O}(\log N)$ 1754b4, 53 lines

```
struct Node {
  Node *l = 0, *r = 0;
  int val, y, c = 1;
  Node(int val) : val(val), y(rand()) {}
  void recalc();
};

int cnt(Node* n) { return n ? n->c : 0; }
void Node::recalc() { c = cnt(l) + cnt(r) + 1;
    }

template<class F> void each(Node* n, F f) {
  if (n) { each(n->l, f); f(n->val); each(n->r
      , f); }
}

pair<Node*, Node*> split(Node* n, int k) {
  if (!n) return {};
  if (cnt(n->l) >= k) { // "n->val >= k" for
      lower_bound(k)
    auto [L,R] = split(n->l, k);
    n->l = R;
    n->recalc();
    return {L, n};
  } else {
    auto [L,R] = split(n->r,k - cnt(n->l) - 1)
        ; // and just "k"
```

```
    n->r = L;
    n->recalc();
    return {n, R};
  }
}

Node* merge(Node* l, Node* r) {
  if (!l) return r;
  if (!r) return l;
  if (l->y > r->y) {
    l->r = merge(l->r, r);
    return l->recalc(), l;
  } else {
    r->l = merge(l, r->l);
    return r->recalc(), r;
  }
}

Node* ins(Node* t, Node* n, int pos) {
  auto [l,r] = split(t, pos);
  return merge(merge(l, n), r);
}

// Example application: move the range [l, r)
    to index k
void move(Node*& t, int l, int r, int k) {
  Node *a, *b, *c;
  tie(a,b) = split(t, l); tie(b,c) = split(b,
      r - l);
  if (k <= l) t = merge(ins(a, b, k), c);
  else t = merge(a, ins(c, b, k - r));
}
```

FenwickTree.h

Description: `update(i,x): a[i] += x;` `query(i): sum in [0, i);` `lower_bound(sum): min pos st sum of [0, pos] >= sum, returns n if all < sum, or -1 if empty sum.`

Time: Both operations are $\mathcal{O}(\log N)$. f74d01, 16 lines

```
struct FT {
  int n; V<ll> s;
  FT(int _n) : n(_n), s(_n) {}
  void update(int i, ll x) {
    for (; i < n; i |= i + 1) s[i] += x; }
  ll query(int i, ll r = 0) {
    for (; i > 0; i &= i - 1) r += s[i-1];
        return r; }
  int lower_bound(ll sum) {
    if (sum <= 0) return -1; int pos = 0;
    for (int pw = 1 << __lg(n); pw; pw >>= 1){
      if (pos+pw <= n && s[pos + pw-1] < sum)
        pos += pw, sum -= s[pos-1];
    }
    return pos;
  }
}; // Hash = d05c4f without lower_bound
```

FenwickTreeRange.h

Description: Range add Range sum with FT.

Time: Both operations are $\mathcal{O}(\log N)$. 8fc549, 11 lines

```
FT f1(n), f2(n);
// a[l...r] += v; 0 <= l <= r < n
auto upd = [&](int l, int r, ll v) {
  f1.update(l, v), f1.update(r + 1, -v);
  f2.update(l, v*(l-1)), f2.update(r+1, -v*r);
}; // a[l] + ... + a[r]; 0 <= l <= r < n
auto sum = [&](int l, int r) { ++r;
  ll sub = f1.query(l) * (l-1) - f2.query(l);
  ll add = f1.query(r) * (r-1) - f2.query(r);
  return add - sub;
};
```

FenwickTree2d.h

Description: Computes sums a[i,j] for all i<I, j<J, and increases single elements a[i,j]. Requires that the elements to be updated are known in advance (call fakeUpdate() before init()).

Time: $\mathcal{O}\left(\log^2 N\right)$. (Use persistent segment trees for $\mathcal{O}(\log N)$.)

"FenwickTree.h" d53ef2, 20 lines

```
struct FT2 {
  V<vi> ys; V<FT> ft;
  FT2(int limx) : ys(limx) {}
  void fakeUpdate(int x, int y) {
    for (;x<sz(ys);x|=x+1) ys[x].push_back(y);
  }
  void init() { for (vi& v : ys)
    sort(all(v)), ft.emplace_back(sz(v));
  }
  int ind(int x, int y) {
    return (int)(lower_bound(all(ys[x]), y) -
        ys[x].begin()); }
  void update(int x, int y, ll dif) {
    for (; x < sz(ys); x |= x + 1)
      ft[x].update(ind(x, y), dif);
  }
  ll query(int x, int y) { ll sum = 0;
    for (; x; x &= x - 1)
      sum += ft[x-1].query(ind(x-1, y));
    return sum;
} };
```

RMQ.h

Description: Range Minimum Queries on an array. Returns min(V[a], V[a + 1], ... V[b - 1]) in constant time.

Usage: `RMQ rmq(values);` `rmq.query(inclusive, exclusive);`

Time: $\mathcal{O}(|V|\log|V| + Q)$ 7d2211, 15 lines

```
template<class T>
struct RMQ {
  V<V<T>> jmp;
  RMQ(const V<T>& V) : jmp(1, V) {
    for (int pw = 1, k = 1; pw * 2 <= sz(V);
        pw *= 2, ++k) {
      jmp.emplace_back(sz(V) - pw * 2 + 1);
      rep(j,0,sz(jmp[k]))
        jmp[k][j] = min(jmp[k - 1][j], jmp[k -
            1][j + pw]);
    }
  }
  T query(int a, int b) {
    assert(a < b); // or return inf if a == b
    int dep = 31 - __builtin_clz(b - a);
    return min(jmp[dep][a], jmp[dep][b - (1 <<
        dep)]);
} };
```

MoTree.h

Description: Build Euler tour of $2N$ size - write node at first enter and last exit. Now, $\texttt{Path}(u,v)$ with $in[u] < in[v]$ is a segment. If $\texttt{lca}(u,v) = u$ then it is $[in[u], in[v]]$. Otherwise it is $[out[u], in[v]]$ + `LCA node`. Nodes that appear exactly once in each segment are relevant, ignore others, handle LCA separately.

Time: $\mathcal{O}\left(Q\sqrt{N}\right)$

MoUpdate.h

Description: Set block size $B = (2n^2)^{1/3}$. Sort queries by $(\lfloor\frac{L}{B}\rfloor, \lfloor\frac{R}{B}\rfloor, t)$, where t = number of updates before this query. Then process queries in sorted order, modify L, R and then apply/undo the updates to answer.

Time: $\mathcal{O}\left(Bq + qn^2/B^2\right)$ or $\mathcal{O}\left(qn^{2/3}\right)$ with that B.