Financial University under the Government of Russian Federation

Viktor Chagai

Semyon Babich, Damir Bekirov, Yury Korobkov

Northern Eurasia Finals 2024

December 15, 2024

Data Structures (1)

fenwick.hh

Description: Fenwick my twizz=)

22 lines

```
struct FT {
  vector<ll> s;
  FT(int n) : s(n) {}
  void update(int pos, ll dif) { // a[pos] += dif
    for (; pos < sz(s); pos |= pos + 1) s[pos] += dif;
  }
  ll query(int pos) { // sum of values in [0, pos)
    ll res = 0;
    for (; pos > 0; pos &= pos - 1) res += s[pos-1];
    return res;
  }
  int lower_bound(ll sum) {// min pos st sum of [0, pos] >= sum
    // Returns n if no sum is >= sum, or -1 if empty sum is.
    if (sum <= 0) return -1;
    int pos = 0;
    for (int pw = 1 << 25; pw; pw >>= 1) {
      if (pos + pw <= sz(s) && s[pos + pw-1] < sum)
        pos += pw, sum -= s[pos-1];
    }
    return pos;
  }
};
```

segmentTree.hh

Description: Na otrezke mozhno gryaz delat

58 lines

```
template <typename T>
struct SegmentTree{
public:
    SegmentTree(vector <T>& _a) : n(_a.size()), a(_a) : {
        t.assign(4 * n, 0);
        mod.assign(4 * n, 0);
        build(1, 0, n);
    }
    void update(int v = 1, int tl = 0, int tr = n, int l, int r
        , T x){
        if (l >= r || tl >= tr) return;
        if (l == tl && r == tr){
            apply(v, l, r, x);
        }
        else {
            push(v, tl, tr);
            int mid = (tl + tr) >> 1;
            update(2 * v, tl, mid, l, min(r, mid), x);
            update(2 * v + 1, mid, tr, max(l, mid), r, x);
            t[v] = min(t[2 * v], t[2 * v + 1]);
        }
    }
    ll get_min(int v = 1, int tl = 0, int tr = n, int l, int r)
        {
        if (l >= r || tl >= tr) return INFLL;
        if (l == tl && r == tr){
            return t[v];
        }
        else {
            push(v, tl, tr);
            int mid = (tl + tr) >> 1;
            return min(get_min(2 * v, tl, mid, l, min(r, mid)),
                   get_min(2 * v + 1, mid, tr, max(l, mid), r));
        }
    }
private:
    int n;
    vector <T> &a;
    vector <T> t, mod;
    void build(int v, int l, int r){
        if (l == r - 1) {
            t[v] = a[l];
        }
        else {
            int mid = (l + r) >> 1;
            build(2 * v, l, mid);
            build(2 * v + 1, mid, r);
            t[v] = min(t[2 * v], t[2 * v + 1]);
        }
    }
    void apply (int v, int l, int r, ll x){
        t[v] += x;
        mod[v] += x;
    }
    void push (int v, int l, int r){
        int mid = (l + r) >> 1;
        apply(2*v, l, mid, mod[v]);
        apply(2*v + 1, mid, r, mod[v]);
        mod[v] = 0;
    }
};
```

sparseTable.hh

Description: Geek from Tyumen Region thinks that he is RMQ Data Structre

27 lines

```
template <typename T>
struct SparseTable{
public:
    SparseTable (vector <T> &_a) : n(_a.size()), a(_a) {
        init(n);
    }
    T rmq(T l, T r) {
        T t = __lg(r - l);
        return min(g[t][l], g[t][r - (1 << t)]);
    }
private:
    int n;
    vector <T> &a;
    vector <vector <T>> g;
    void init(int n) {
        int logn = __lg(n);
        g.assign(logn + 1, vector <T>(n));
        for (int i = 0; i < n; ++i) {
            g[0][i] = a[i];
        }
        for (int l = 0; l <= logn - 1; l++) {
            for (int i = 0; i + (2 << l) <= n; i++) {
                g[l + 1][i] = min(g[l][i], g[l][i + (1 << l)]);
            }
        }
    }
};
```

treap.hh

Description: Just treap=)

55 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 pa = split(n->l, k);
    n->l = pa.second;
    n->recalc();
    return {pa.first, n};
  } else {
    auto pa = split(n->r, k - cnt(n->l) - 1); // and just "k"
    n->r = pa.first;
    n->recalc();
    return {n, pa.second};
  }
}

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);
    l->recalc();
    return l;
  } else {
    r->l = merge(l, r->l);
    r->recalc();
    return 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));
}
```

lazySegmentTree.hh

Description: Segment tree with ability to add or set values of large intervals, and compute max of intervals.
Usage: `Node* tr = new Node(v, 0, sz(v));`
Time: $\mathcal{O}(\log N)$.

50 lines

```
const int inf = 1e9;
struct Node {
  Node *l = 0, *r = 0;
  int lo, hi, mset = inf, madd = 0, val = -inf;
```

```
  Node(int lo,int hi) : lo(lo), hi(hi) {} // Large interval of
      -inf
  Node(vector <int>& v, int lo, int hi) : lo(lo), hi(hi) {
    if (lo + 1 < hi) {
      int mid = lo + (hi - lo)/2;
      l = new Node(v, lo, mid); r = new Node(v, mid, hi);
      val = max(l->val, r->val);
    }
    else val = v[lo];
  }
  int query(int L, int R) {
    if (R <= lo || hi <= L) return -inf;
    if (L <= lo && hi <= R) return val;
    push();
    return max(l->query(L, R), r->query(L, R));
  }
  void set(int L, int R, int x) {
    if (R <= lo || hi <= L) return;
    if (L <= lo && hi <= R) mset = val = x, madd = 0;
    else {
      push(), l->set(L, R, x), r->set(L, R, x);
      val = max(l->val, r->val);
    }
  }
  void add(int L, int R, int x) {
    if (R <= lo || hi <= L) return;
    if (L <= lo && hi <= R) {
      if (mset != inf) mset += x;
      else madd += x;
      val += x;
    }
    else {
      push(), l->add(L, R, x), r->add(L, R, x);
      val = max(l->val, r->val);
    }
  }
  void push() {
    if (!l) {
      int mid = lo + (hi - lo)/2;
      l = new Node(lo, mid); r = new Node(mid, hi);
    }
    if (mset != inf)
      l->set(lo, hi, mset), r->set(lo, hi, mset), mset = inf;
    else if (madd)
      l->add(lo, hi, madd), r->add(lo, hi, madd), madd = 0;
  }
};
```

persistentDSU.hh

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: $\mathcal{O}(\log(N))$ 21 lines

```
struct DSU {
  vector <int> e; vector <pair <int, int>> st;
  DSU (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;
  }
};
```

intervalContainer.hh

Description: Add and remove intervals from a set of disjoint intervals. Will merge the added interval with any overlapping intervals in the set when adding. Intervals are [inclusive, exclusive).
Time: $\mathcal{O}(\log N)$ 25 lines

```
using pii = pair <int, int>;

set <pii>::iterator addInterval(set <pii>& is, int L, int R) {
  if (L == R) return is.end();
  auto it = is.lower_bound({L, R}), before = it;
  while (it != is.end() && it->first <= R) {
    R = max(R, it->second);
    before = it = is.erase(it);
  }
  if (it != is.begin() && (--it)->second >= L) {
    L = min(L, it->first);
    R = max(R, it->second);
    is.erase(it);
  }
  return is.insert(before, {L,R});
}

void removeInterval(set <pii>& is, int L, int R) {
  if (L == R) return;
  auto it = addInterval(is, L, R);
  auto r2 = it->second;
  if (it->first == L) is.erase(it);
  else (int&)it->second = L;
  if (R != r2) is.emplace(R, r2);
}
```

Mathematics (2)

2.1 Master Theorem

$$T(n) = \begin{cases} aT(\frac{n}{b}) + \Theta(n^c) & n > n_0 \\ \Theta(1) & n \le n_0 \end{cases}$$

- $c > \log_b a : T(n) = \Theta(n^c)$
- $c = \log_b a : T(n) = \Theta(n^c \log n)$
- $c < \log_b a : T(n) = \Theta(n^{\log_b a})$

2.2 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.

2.3 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

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

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

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

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

2.5 Geometry

2.5.1 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: $\dfrac{a+b}{a-b} = \dfrac{\tan\dfrac{\alpha+\beta}{2}}{\tan\dfrac{\alpha-\beta}{2}}$

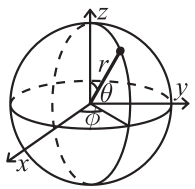
2.5.2 Quadrilaterals

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

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

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



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

2.6 Derivatives/Integrals

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

Integration by parts:

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

2.7 Sums

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

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$
$$1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(2n+1)(n+1)}{6}$$
$$1^3 + 2^3 + 3^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$$
$$1^4 + 2^4 + 3^4 + \cdots + n^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

2.8 Series

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

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,

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

2.10 Generating functions

Example of finding generating functions:

$a_0 = 1$

$a_1 = 2$

$a_n = 6a_{n-1} - 8a_{n-2} + n, n \ge 2$

$$G(z) = a_0 + a_1 z + \sum_{n=2}^{\infty}(6a_{n-1} - 8a_{n-2} + n)z^n$$

$$G(z) = a_0 + a_1 z + 6\sum_{n=2}^{\infty} a_{n-1}z^n - 8\sum_{n=2}^{\infty} a_{n-2}z^n + \sum_{n=2}^{\infty} nz^n$$

$$G(z) = a_0 + a_1 z + 6z\sum_{n=1}^{\infty} a_n z^n - 8z^2\sum_{n=0}^{\infty} a_n z^n + \sum_{n=2}^{\infty} nz^n$$

$$G(z) = a_0 + a_1 z + 6z(G(z) - a_0) - 8z^2G(z) + \sum_{n=2}^{\infty} nz^n$$

$$G(z) = 1 - 4z + 6zG(z) - 8z^2G(z) + \sum_{n=2}^{\infty} nz^n$$

To calculate $\sum_{n=2}^{\infty} nz^n$ use generating function $B(z)$ for $b_n = (1, 1, 1, 1, ...)$:

$zB'(z) = z(\sum_{n=0}^{\infty} b_n z^n)' = \sum_{n=0}^{\infty} b_n z^n$

Then we can do:

$\sum_{n=2}^{\infty} nz^2 = z\sum_{n=2}^{\infty} nz^{n-1} = z(\sum_{n=2}^{\infty} z^n)'$

$\sum_{n=2}^{\infty} z^n = \sum_{n=0}^{\infty} z^n - 1 - z = \frac{1}{1-z} - 1 - z = \frac{z^2}{1-z}$

$z(\frac{z^2}{1-z})' = \frac{z^2(2-z)}{(1-z)^2}$

Then:

$G(z) = 1 - 4z + 6zG(z) - 8z^2 2G(z) + \frac{z^2(2-z)}{(1-z)^2}$

$G(z) = \frac{1-6z+11z^2-5z^3}{(1-6z+8z^2)(1-z)^2}$

Последовательность	Производящая функция в виде ряда	Производящая функция в замкнутом виде
$(1, 0, 0, \ldots)$	1	1
$(0, 0, \ldots, 0, 1, 0, 0 \ldots)$ (m нулей в начале)	z^m	z^m
$(1, 1, 1, \ldots)$	$\sum z^n$	$\frac{1}{1-z}$
$(1, 0, 0, \ldots, 0, 1, 0, 0, \ldots 0, 1, 0, 0 \ldots)$ (повторяется через m)	$\sum z^{nm}$	$\frac{1}{1-z^m}$
$(1, -1, 1, -1, \ldots)$	$\sum(-1)^n z^n$	$\frac{1}{1+z}$
$(1, 2, 3, 4, \ldots)$	$\sum(n+1)z^n$	$\frac{1}{(1-z)^2}$
$(1, 2, 4, 8, 16, \ldots)$	$\sum 2^n z^n$	$\frac{1}{(1-2z)}$
$(1, r, r^2, r^3, \ldots)$	$\sum r^n z^n$	$\frac{1}{(1-rz)}$
$(\binom{m}{0}, \binom{m}{1}, \binom{m}{2}, \binom{m}{3}, \ldots)$	$\sum \binom{m}{n} z^n$	$(1+z)^m$
$(1, \binom{m}{m}, \binom{m+1}{m}, \binom{m+2}{m}, \ldots)$	$\sum \binom{m+n-1}{n} z^n$	$\frac{1}{(1-z)^m}$
$(1, \binom{m+1}{m}, \binom{m+2}{m}, \binom{m+3}{m}, \ldots)$	$\sum \binom{m+n}{n} z^n$	$\frac{1}{(1-z)^{m+1}}$
$(0, 1, -\frac{1}{2}, \frac{1}{3}, -\frac{1}{4}, \ldots)$	$\sum \frac{(-1)^{n+1}}{n} z^n$	$\ln(1+z)$
$(1, 1, \frac{1}{2}, \frac{1}{6}, \frac{1}{24}, \ldots)$	$\sum \frac{1}{n!} z^n$	e^z
$(1, -\frac{1}{2!}m^2, \frac{1}{4!}m^4, -\frac{1}{6!}m^6, \frac{1}{8!}m^8, \ldots)$	$\sum \frac{1}{(2n)!} m^{(2n)}$	$\cos m$
$(m, -\frac{1}{3!}m^3, \frac{1}{5!}m^5, -\frac{1}{7!}m^7, \frac{1}{9!}m^9, \ldots)$	$\sum \frac{1}{(2n-1)!} m^{(2n-1)}$	$\sin m$

DP (3)

3.1 Layer DP Optimization

divideAndConquer.hh

Description: when $opt[i][j] \le opt[i][j+1]$
Time: $\mathcal{O}(kn \log n)$

13 lines

```
void calc(int tl, int tr, int l, int r, int layer){
    if(tl > tr) return;
    int mid = (tl + tr) >> 1;
    int opt = -1;
    for (int i = l; i <= min(r, mid - 1); ++i) {
        if (dp[mid][layer + 1] > dp[i][layer] + cost[i + 1][mid
            ]) {
            opt = i;
            dp[mid][layer + 1] = dp[i][layer] + cost[i + 1][mid
                ];
        }
    }
    calc(tl, mid - 1, l, opt, layer);
    calc(mid + 1, tr, opt, r, layer);
}
```

knuthOptimization.hh

Description: when $opt[i-1][j] \le opt[i][j] \le opt[i][j+1]$
Time: $\mathcal{O}(n^2)$

11 lines

```
for (int len = 2; len <= n; ++len) {
    for (int l = 0; l <= n - len; ++l) {
        int r = l + len;
        for (int i = cut[l][r - 1]; i <= cut[l + 1][r]; ++i) {
            if (dp[l][r] > cost[l + 1][i] + dp[l][i] + cost[i +
                2][r] + dp[i + 1][r]) {
                cut[l][r] = i;
                dp[l][r] = cost[l + 1][i] + dp[l][i] + cost[i +
                    2][r] + dp[i + 1][r];
            }
        }
    }
}
```

convexHullTrick.hh

Description: Use this if your dp transformable to: $dp[i][m] = min(a[k] * x + b[k])$
E.g. $f[i][j] = min(f[k][j-1] + (x_{i-1} - x_k)^2)$
$f[i][j] = min(f[k][j-1] + x_k^2 - 2x_k x_{i-1}) + x_{i-1}^2$
$a[k] = f[k][j-1] + x_k^2, b[k] = -2x_k$
Time: $\mathcal{O}(nk)$

35 lines

```
struct Line {
    mutable ll k, m, p; // p is the position from which the
        line is optimal
    ll val (ll x) const { return k * x + m; }
    bool operator< (const Line& o) const { return p < o.p; }
};
ll floordiv (ll a, ll b) {
    return a / b - ((a^b) < 0 && a % b);
}
// queries and line intersections should be in range (-INF, INF
    )
const ll INF = 1e17;
struct LineContainer : vector<Line> {
    ll isect(const Line& a, const Line& b) {
        if (a.k == b.k) return a.m > b.m ? (-INF) : INF;
        ll res = floordiv(b.m - a.m, a.k - b.k);
        if (a.val(res) < b.val(res)) res++;
        return res;
    }
    void add(ll k, ll m) {
        Line a = {k,m,INF};
        while (!empty() && isect(a, back()) <= back().p)
            pop_back();
        a.p = empty() ? (-INF) : isect(a, back());
        push_back(a);
    }
    ll query(ll x) {
        assert(!empty());
        return (--upper_bound(begin(), end(), Line({0, 0, x})))
            ->val(x);
    }
    int qi = 0;
    ll sorted_query(ll x) {
        assert(!empty());
        qi = min(qi, (int)size() - 1);
        while(qi < size() - 1 && (*this)[qi + 1].p <= x) qi++;
        return (*this)[qi].val(x);
    }
};
```

liChaoTree.hh

Description: good with lazySegmentTree.hh idea sometimes. It's just got to be here. Don't touch it.

49 lines

```
struct LiChaoTree{
    struct line {
        int k = 0, m = 0;
        line() {}
        line(int k, int m): k(k), m(m) {}
        int get(int x) {
            return k * x + m;
        }
    };
public:
    LiChaoTree (int _maxn) : maxn(_maxn) {
        t.assign(4 * maxn);
    }
    void upd(int v = 1, int tl = 0, int tr = maxn - 1, line L)
        {
        if (tl > tr) {
            return;
        }
        int tm = (tl + tr) / 2;
        bool l = L.get(tl) > t[v].get(tl);
        bool mid = L.get(tm) > t[v].get(tm);
        if (mid) {
            swap(L, t[v]);
        }
        if (l != mid) {
            upd(2 * v, tl, tm - 1, L);
        }
        else {
            upd(2 * v + 1, tm + 1, tr, L);
        }
    }
    int get(int v = 1, int tl = 0, int tr = maxn - 1, int x) {
        if (tl > tr) {
            return 0;
        }
        int tm = (tl + tr) / 2;
        if (x == tm) {
            return t[v].get(x);
        }
        if (x < tm) {
            return max(t[v].get(x), get(2 * v, tl, tm - 1, x));
        }
        else {
            return max(t[v].get(x), get(2 * v + 1, tm + 1, tr,
                x));
        }
    }
private:
    int maxn;
    vector <line> t;
}
```

lambdaOptimization.hh

Description: Transforming CHT to: to $dp[i] = min(dp[j] + cost(j+1, i)) + \lambda$
Using binary search for λ to find first that best number of segments is exactly k
Time: $\mathcal{O}(n \log n)$

31 lines

```
void init() {
    for (int i = 0; i < maxn; i++) {
        dp[i] = make_pair(inf, 0);
    }
}
pair <ll, int> check(ll x) { // change this to CHT
    init();
    dp[0] = make_pair(0ll, 0); // 1-indexation
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j < i; j++) {
            dp[i] = min(dp[i], {dp[j].first + cost[j + 1][i] +
                x, dp[j].second + 1});
        }
    }
    return dp[n];
}
ll solve() {
    ll l = -1e14;
    ll r = 1;
    while (l + 1 < r) {
        ll mid = (l + r) / 2;
        pair<ll, int> x = check(mid);
        if (x.second >= k) {
            l = mid;
        }
        else {
            r = mid;
        }
    }
    pair<ll, int> result = check(l);
    return result.first - l * return.second;
}
```

3.2 Simulated Annealing

simulatedAnnealing.hh

Description: Use it if you chill guy =)
There is solution of a problem of placing 8 queens on a chessboard in a such way that they don't attack each other
$f(p)$ - number of the queens that don't attack anothers

40 lines

```
const int n = 100;
const int k = 1000;

int f(vector<int> &p) {
    int s = 0;
    for (int i = 0; i < n; i++) {
        int d = 1;
        for (int j = 0; j < i; j++)
            if (abs(i - j) == abs(p[i] - p[j]))
                d = 0;
        s += d;
    }
    return s;
}

double rnd() { return double(rand()) / RAND_MAX; }

int main() {
    vector<int> v(n);
```

```
    iota(v.begin(), v.end(), 0);
    shuffle(v.begin(), v.end()); // generate initial
        permutation
    int ans = f(v);

    double t = 1;
    for (int i = 0; i < k && ans < n; i++) {
        t *= 0.99;
        vector<int> u = v;
        swap(u[rand() % n], u[rand() % n]);
        int val = f(u);
        if (val > ans || rnd() < exp((val - ans) / t)) {
            v = u;
            ans = val;
        }
    }

    for (int x : v)
        cout << x + 1 << " ";

    return 0;
}
```

Number Theory (4)

bitGCD.hh

Description: Fast GCD
Time: Can speed up __gcd by 1.7x

19 lines

```
unsigned int gcd (unsigned int u, unsigned int v) {
    int shift, uz, vz;
    if (u == 0) return v;
    if (v == 0) return u;
    uz = __builtin_ctz(u);
    vz = __builtin_ctz(v);
    shift = uz > vz ? vz : uz;
    u >>= uz;
    do {
        v >>= vz;
        int diff = v;
        diff -= u;
        vz = __builtin_ctz(diff);
        if (diff == 0) break;
        if (v < u) u = v;
        v = abs(diff);
    } while (1);
    return u << shift;
}
```

euclid.hh

Description: Finds two integers x and y, such that $ax + by = \gcd(a, b)$. If you just need gcd, use the built in __gcd instead. If a and b are coprime, then x is the inverse of $a \pmod b$.

5 lines

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

factor.hh

Description: Pollard-rho randomized factorization algorithm. Returns prime factors of a number, in arbitrary order (e.g. 2299 -> $\{11, 19, 11\}$).
Time: $\mathcal{O}\left(n^{1/4}\right)$, less for numbers with small factors.

"MillerRabin.hh" 18 lines

```
ull pollard(ull n) {
  ull x = 0, y = 0, t = 30, prd = 2, i = 1, q;
  auto f = [&](ull x) { return modmul(x, x, n) + i; };
  while (t++ % 40 || __gcd(prd, n) == 1) {
    if (x == y) x = ++i, y = f(x);
    if ((q = modmul(prd, max(x,y) - min(x,y), n))) prd = q;
    x = f(x), y = f(f(y));
  }
  return __gcd(prd, n);
}
vector<ull> factor(ull n) {
  if (n == 1) return {};
  if (isPrime(n)) return {n};
  ull x = pollard(n);
  auto l = factor(x), r = factor(n / x);
  l.insert(l.end(), all(r));
  return l;
}
```

fastEratosthenes.hh

Description: Prime sieve for generating all primes smaller than LIM.
Time: LIM=1e9 $\approx$ 1.5s

20 lines

```
const int LIM = 1e6;
bitset<LIM> isPrime;
vi eratosthenes() {
  const int S = (int)round(sqrt(LIM)), R = LIM / 2;
  vi pr = {2}, sieve(S+1); pr.reserve(int(LIM/log(LIM)*1.1));
  vector<pii> cp;
  for (int i = 3; i <= S; i += 2) if (!sieve[i]) {
    cp.push_back({i, i * i / 2});
    for (int j = i * i; j <= S; j += 2 * i) sieve[j] = 1;
  }
  for (int L = 1; L <= R; L += S) {
    array<bool, S> block{};
    for (auto &[p, idx] : cp)
      for (int i=idx; i < S+L; idx = (i+=p)) block[i-L] = 1;
    rep(i,0,min(S, R - L))
      if (!block[i]) pr.push_back((L + i) * 2 + 1);
  }
  for (int i : pr) isPrime[i] = 1;
  return pr;
}
```

millerRabin.hh

Description: Deterministic Miller-Rabin primality test. Guaranteed to work for numbers up to $7 \cdot 10^{18}$; for larger numbers, use Python and extend A randomly.
Time: 7 times the complexity of $a^b \mod c$.

12 lines

```
bool isPrime(ull n) {
  if (n < 2 || n % 6 % 4 != 1) return (n | 1) == 3;
  ull A[] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022},
      s = __builtin_ctzll(n-1), d = n >> s;
  for (ull a : A) {   // ^ count trailing zeroes
    ull p = modpow(a%n, d, n), i = s;
    while (p != 1 && p != n - 1 && a % n && i--)
      p = modmul(p, p, n);
    if (p != n-1 && i != s) return 0;
  }
  return 1;
}
```

modLog.hh

Description: Returns the smallest $x > 0$ s.t. $a^x = b \pmod m$, or -1 if no such x exists. modLog(a,1,m) can be used to calculate the order of a.
Time: $\mathcal{O}\left(\sqrt{m}\right)$

11 lines

```
ll modLog(ll a, ll b, ll m) {
  ll n = (ll) sqrt(m) + 1, e = 1, f = 1, j = 1;
  unordered_map<ll, ll> A;
  while (j <= n && (e = f = e * a % m) != b % m)
    A[e * b % m] = j++;
  if (e == b % m) return j;
  if (__gcd(m, e) == __gcd(m, b))
    rep(i,2,n+2) if (A.count(e = e * f % m))
      return n * i - A[e];
  return -1;
}
```

phiCalc.hh

Description: *Euler's* ϕ function is defined as $\phi(n) := \#$ of positive integers $\leq n$ that are coprime with n. $\phi(1) = 1$, p prime $\Rightarrow \phi(p^k) = (p-1)p^{k-1}$, m, n coprime $\Rightarrow \phi(mn) = \phi(m)\phi(n)$. If $n = p_1^{k_1} p_2^{k_2} ... p_r^{k_r}$ then $\phi(n) = (p_1 - 1)p_1^{k_1-1} ... (p_r - 1)p_r^{k_r-1}$. $\phi(n) = n \cdot \prod_{p|n}(1 - 1/p)$.
$\sum_{d|n} \phi(d) = n$, $\sum_{1 \leq k \leq n, \gcd(k,n)=1} k = n\phi(n)/2, n > 1$
Euler's thm: a, n coprime $\Rightarrow a^{\phi(n)} \equiv 1 \pmod n$.
Fermat's little thm: p prime $\Rightarrow a^{p-1} \equiv 1 \pmod p$ $\forall a$.

8 lines

```
const int LIM = 5000000;
int phi[LIM];

void calculatePhi() {
  rep(i,0,LIM) phi[i] = i&1 ? i : i/2;
  for (int i = 3; i < LIM; i += 2) if(phi[i] == i)
    for (int j = i; j < LIM; j += i) phi[j] -= phi[j] / i;
}
```