ICPC Notebook

template

0settings.sh

```
export CXXFLAGS='-O3 -std=c++2a -D_GLIBCXX_DEBUG -
D_GLIBCXX_DEBUG_PEDANTIC -Wfatal-errors'
```

1template.hpp

md5: f368a0

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define pii pair<int, int>
#define pll pair<ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define ov4(a, b, c, d, name, ...) name
#define rep3(i, a, b, c) for(ll i = (a); i < (b); i += (c))
#define rep2(i, a, b) rep3(i, a, b, 1)
#define rep1(i, n) rep2(i, 0, n)
#define rep0(n) rep1(aaaaa, n)
#define rep(...) ov4(__VA_ARGS__, rep3, rep2, rep1, rep0)
(__VA_ARGS__)
#define per(i, a, b) for(ll i = (a)-1; i >= (b); i--)
#define fore(e, v) for(auto&& e : v)
#define all(a) begin(a), end(a)
#define si(a) (int)(size(a))
#define lb(v, x) (lower_bound(all(v), x) - begin(v))
#define eb emplace_back

template<typename T, typename S> bool chmin(T& a, const S& b) {
return a > b ? a = b, 1 : 0; }
template<typename T, typename S> bool chmax(T& a, const S& b) {
return a < b ? a = b, 1 : 0; }

const int INF = 1e9 + 100;
const ll INFL = 3e18 + 100;

#define i128 __int128_t

struct _ {
   _() { cin.tie(0)->sync_with_stdio(0), cout.tie(0); }
} __;
```

hash.sh

```
# 使い方: sh hash.sh -> コピペ -> Ctrl + D
# コメント・空白・改行を削除して md5 でハッシュする
g++ -dD -E -P -fpreprocessed - | tr -d '[:space:]' | md5sum |
cut -c-6
```

random_test.sh

```
# correct wrong generate
make $1
make $2
make $3
while true;do
./$3 > input.txt
c=$(./$1 < input.txt)
w=$(./$2 < input.txt)
if [[ $c = $w ]]; then
echo "pass"
else
echo "WA"
echo "$(cat input.txt)"
echo "o:"
echo "$c"
echo "x:"
echo "$w"
exit
fi
done
```

rnd.hpp

md5: a81b0a

```
ll rnd(ll l, ll r) {  //[l, r)
   static mt19937_64
gen(chrono::steady_clock::now().time_since_epoch().count());
   return uniform_int_distribution<ll>(l, r - 1)(gen);
}
template<typename T> void rndshuf(vector<T>& v) { rep(i, 1,
si(v)) swap(v[i], v[rnd(0, i)]); }
template<class T> vector<T> rvi(int n, T l, T r, bool unique =
false) {
   if(unique) {
      assert(r - l >= n);
      vector<T> res;
      rep(i, n) res.eb(rnd(l, r - n + 1));
      sort(all(res));
      rep(i, n) res[i] += i;
      rndshuf(res);
      return res;
   }
   vector<T> v(n);
   fore(e, v) e = rnd(l, r);
   return v;
}
```

data-structure

BIT.hpp

md5: 5f35a3

```
struct BIT {
   vl a;
   BIT(ll n) : a(n + 1) {}
   void add(ll i, ll x) {
      i++;
      while(i < si(a)) a[i] += x, i += i & -i;
   }
   ll sum(ll r) {
      ll s = 0;
      while(r) s += a[r], r -= r & -r;
      return s;
   }
   ll sum(ll l, ll r) { return sum(r) - sum(l); }
   // minimize i s.t. sum(i) >= w
   int lower_bound(ll w) {
      int x = 0, N = si(a) - 1;
      for(int k = 1 << __lg(N); k; k >>= 1) {
         if(x + k <= N && a[x + k] < w) {
            w -= a[x + k];
            x += k;
         }
      }
      return x + 1;
   }
};
```

FastSet.hpp

md5: 9dd1e2

```
using U = uint64_t;
const U B = 64;
struct FS {
   U n;
   vector<vector<U>> a;
   FS(U n) : n(n) {
      do a.eb(n = (n + B - 1) / B);
      while(n > 1);
   }
   bool operator[](ll i) const { return a[0][i / B] >> (i % B)
& 1; }
   void set(ll i) {
      for(auto& v : a) {
         v[i / B] |= 1ULL << (i % B);
         i /= B;
      }
   }
   void erase(ll i) {
      for(auto& v : a) {
         v[i / B] &= ~(1ULL << (i % B));
         if(v[i / B]) break;
         i /= B;
      }
   }
   ll next(ll i) {
      rep(h, si(a)) {
         i++;
         if(i / B >= si(a[h])) break;
         U d = a[h][i / B] >> (i % B);
         if(d) {
            i += countr_zero(d);
            while(h--) i = i * B + countr_zero(a[h][i]);
            return i;
         }
         i /= B;
      }
      return n;
   }
   ll prev(ll i) {
      rep(h, si(a)) {
         i--;
         if(i < 0) break;
         U d = a[h][i / B] << (~i % B);
         if(d) {
            i -= countl_zero(d);
            while(h--) i = i * B + __lg(a[h][i]);
            return i;
         }
         i /= B;
      }
      return -1;
   }
};
```

Skew-Heap.hpp

md5: 38dad3

```
template<typename T, bool isMin = true> struct SkewHeap {
   struct Node {
      T key, laz;
      Node *l, *r;
      int idx;
      Node() = default;
      Node(const T& k, int i = -1) : key(k), laz(0),
l(nullptr), r(nullptr), idx(i) {}
   };
   using P = Node*;
   static void propagate(P x) {
      if(x->laz == 0) return;
      if(x->l) x->l->laz += x->laz;
      if(x->r) x->r->laz += x->laz;
      x->key += x->laz;
      x->laz = 0;
   }
   static P meld(P x, P y) {
      if(!x || !y) return x ? x : y;
      if(!comp(x, y)) swap(x, y);
      propagate(x);
      x->r = meld(x->r, y);
      swap(x->l, x->r);
      return x;
   }
   static P alloc(const T& key, int idx = -1) { return new
Node(key, idx); }
   static P pop(P x) {
      propagate(x);
      return meld(x->l, x->r);
   }
   static P push(P x, const T& key, int idx = -1) { return
meld(x, alloc(key, idx)); }
   static void apply(P x, const T& laz) {
      x->laz += laz;
      propagate(x);
   }

   private:
   static inline bool comp(P x, P y) {
      if constexpr(isMin) {
         return x->key + x->laz < y->key + y->laz;
      } else {
         return x->key + x->laz > y->key + y->laz;
      }
   }
};
```

cht.hpp

md5: a05621

```
template<bool isMin = true> struct CHT {
#define x first
#define y second
   CHT() = default;
   deque<pll> v;
   bool empty() { return v.empty(); }
   void clear() { return v.clear(); }
   inline int sgn(ll x) { return !x ? 0 : (x < 0 ? -1 : 1); }
   using D = long double;
   inline bool check(const pll& a, const pll& b, const pll& c)
{
      if(b.y == a.y or c.y == b.y) return sgn(b.x - a.x) *
sgn(c.y - b.y) >= sgn(c.x - b.x) * sgn(b.y - a.y);
      return D(b.x - a.x) * sgn(c.y - b.y) / D(abs(b.y - a.y))
>= D(c.x - b.x) * sgn(b.y - a.y) / D(abs(c.y - b.y));
   }
   void add(ll a, ll b) {
      if(!isMin) a *= -1, b *= -1;
      pll line(a, b);
      if(empty()) v.emplace_front(line);
      else {
         if(ll c = v[0].x; c <= a) {
            if(c == a) {
               if(v[0].y <= b) return;
               v.pop_front();
            }
            while(si(v) >= 2 and check(line, v[0], v[1]))
v.pop_front();
            v.emplace_front(line);
         } else {
            assert(a <= v.back().x);
            if(v.back().x == a) {
               if(v.back().y <= b) return;
               v.pop_back();
            }
            while(si(v) >= 2 and check(v[si(v) - 2], v.back(),
line)) v.pop_back();
            v.emplace_back(line);
         }
      }
   }
   ll get_y(const pll& a, const ll& x) { return a.x * x + a.y;
}
   ll query(ll x) {
      assert(!empty());
      int l = -1, r = si(v) - 1;
      while(l + 1 < r) {
         int m = (l + r) >> 1;
         if(get_y(v[m], x) >= get_y(v[m + 1], x)) l = m;
         else r = m;
      }
      return get_y(v[r], x) * (isMin ? 1 : -1);
   }
   ll query_monotone_inc(ll x) {
      assert(!empty());
      while(si(v) >= 2 and get_y(v[0], x) >= get_y(v[1], x))
v.pop_front();
      return get_y(v[0], x) * (isMin ? 1 : -1);
   }
   ll query_monotone_dec(ll x) {
      assert(!empty());
      while(si(v) >= 2 and get_y(v.back(), x) >= get_y(v.end()
[-2], x)) v.pop_back();
      return get_y(v.back(), x) * (isMin ? 1 : -1);
   }
#undef x
#undef y
};
```

hash_map.hpp

md5: 1893ff

```
#include <bits/extc++.h>
struct chash {
   const uint64_t C = (ll)(4e18 * acos(0)) | 71;
   ll operator()(ll x) const { return __builtin_bswap64(x * C);
}
};
using namespace __gnu_pbds;
template<class T, class S> using hash_map = gp_hash_table<T, S,
chash>;
```

lazy-segtree.hpp

md5: e79596

```
using U = uint64_t;
template<class S, S (*op)(S, S), S (*e)(), class F, S (*mpp)(F,
S), F (*cmpo)(F, F), F (*id)()> struct lazy_segtree {
   lazy_segtree() : lazy_segtree(0) {}
   explicit lazy_segtree(int n) : lazy_segtree(vector<S>(n,
e())) {}
   explicit lazy_segtree(const vector<S>& v) : n(si(v)) {
      s = bit_ceil(U(n));
      log = countr_zero(U(s));
      d = vector<S>(2 * s, e());
      lz = vector<F>(s, id());
      rep(i, n) d[s + i] = v[i];
      per(i, s, 1) update(i);
   }
   void set(int p, S x) {
      p += s;
      PUSH(p);
      d[p] = x;
      rep(i, 1, log + 1) update(p >> i);
   }
   S get(int p) {
      p += s;
      PUSH(p);
      return d[p];
   }
   S prod(int l, int r) {
      if(l == r) return e();
      l += s, r += s;
      per(i, log + 1, 1) {
         if(((l >> i) << i) != l) push(l >> i);
         if(((r >> i) << i) != r) push((r - 1) >> i);
      }
      S sml = e(), smr = e();
      while(l < r) {
         if(l & 1) sml = op(sml, d[l++]);
         if(r & 1) smr = op(d[--r], smr);
         l >>= 1, r >>= 1;
      }
      return op(sml, smr);
   }
   S all_prod() { return d[1]; }
   void apply(int p, F f) {
      // assert(0 <= p && p < n);
      p += s;
      PUSH(p);
      d[p] = mpp(f, d[p]);
      rep(i, 1, log + 1) update(p >> i);
   }
   void apply(int l, int r, F f) {
      // assert(0 <= l && l <= r && r <= _n);
      if(l == r) return;
      l += s, r += s;

      per(i, log + 1, 1) {
         if(((l >> i) << i) != l) push(l >> i);
         if(((r >> i) << i) != r) push((r - 1) >> i);
      }
      int ml = l, mr = r;
      while(l < r) {
         if(l & 1) all_apply(l++, f);
         if(r & 1) all_apply(--r, f);
         l >>= 1, r >>= 1;
      }
      l = ml, r = mr;
      rep(i, 1, log + 1) {
         if(((l >> i) << i) != l) update(l >> i);
         if(((r >> i) << i) != r) update((r - 1) >> i);
      }
   }
   template<class G> int max_right(int l, G g) {
      assert(g(e()));
      if(l == n) return n;
      l += s;
      PUSH(l);
      S sm = e();
      do {
         while(~l & 1) l >>= 1;
         if(!g(op(sm, d[l]))) {
            while(l < s) {
               push(l);
               l <<= 1;
               if(g(op(sm, d[l]))) {
                  sm = op(sm, d[l]);
                  l++;
               }
            }
            return l - s;
         }
         sm = op(sm, d[l]);
         l++;
      } while((l & -l) != l);
      return n;
   }
   template<class G> int min_left(int r, G g) {
      assert(g(e()));
      if(r == 0) return 0;
      r += s;
      PUSH(r - 1);
      S sm = e();
      do {
         r--;
         while(r > 1 && r & 1) r >>= 1;
         if(!g(op(d[r], sm))) {
            while(r < s) {
               push(r);
               r = (2 * r + 1);
               if(g(op(d[r], sm))) {
                  sm = op(d[r], sm);
                  r--;
               }
            }
            return r + 1 - s;
         }
         sm = op(d[r], sm);
      } while((r & -r) != r);
      return 0;
   }
   S operator[](int k) { return get(k); }
   int len() { return n; }

   private:
   int n, s, log;
   vector<S> d;
   vector<F> lz;
   void update(int k) { d[k] = op(d[2 * k], d[2 * k + 1]); }
   void all_apply(int k, F f) {
      d[k] = mpp(f, d[k]);
      if(k < s) lz[k] = cmpo(f, lz[k]);
   }
   void push(int k) {
      all_apply(2 * k, lz[k]);
      all_apply(2 * k + 1, lz[k]);
      lz[k] = id();
   }
   void PUSH(int k) { per(i, log + 1, 1) push(k >> i); }
};
```

li-chao.hpp

md5: ca57d5

```
struct lctree {
   struct line {
      ll a, b;
      line() : a(0), b(INFL) {}
      line(ll a, ll b) : a(a), b(b) {}
      ll get(ll x) { return a * x + b; }
      inline bool over(line r, ll x) { return get(x) <
r.get(x); }
   };
   int n;

   vector<ll> x;
   vector<line> seg;
   lctree() {}
   lctree(const vector<ll>& _x) : x(_x) {
      sort(all(x));
      int n2 = si(x);
      n = 1;
      while(n < n2) n <<= 1;
      x.resize(n);
      rep(i, n2, n) x[i] = x[n2 - 1];
      seg = vector<line>(n * 2);
   }
   void upd(line L, int i, int l, int r) {
      while(true) {
         int mid = l + r >> 1;
         bool lov = L.over(seg[i], x[l]);
         bool rov = L.over(seg[i], x[r - 1]);
         if(lov == rov) {
            if(lov) swap(seg[i], L);
            return;
         }
         bool mov = L.over(seg[i], x[mid]);
         if(mov) swap(seg[i], L);
         if(lov != mov) {
            i = (i << 1), r = mid;
         } else {
            i = (i << 1) + 1, l = mid;
         }
      }
   }
   void upd(line L, unsigned i) {
      int ub = bit_width(i) - 1;
      int l = (n >> ub) * (i - (1 << ub));
      int r = l + (n >> ub);
      upd(L, i, l, r);
   }
   void update(ll a, ll b) { upd(line(a, b), 1, 0, n); }
   void update_segment(ll l, ll r, ll a, ll b) {
      l = lb(x, l) + n, r = lb(x, r) + n;
      line L(a, b);
      for(; l < r; l >>= 1, r >>= 1) {
         if(l & 1) upd(L, l++);
         if(r & 1) upd(L, --r);
      }
   }
   ll query(ll t) {
      ll k = lb(x, t);
      k += n;
      ll res = seg[k].get(t);
      while(k > 1) {
         k >>= 1;
         chmin(res, seg[k].get(t));
      }
      return res;
   }
};
```

line_container.hpp

md5: b018d9

```
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; }
};

template<bool ismin = true> struct LineContainer :
multiset<Line, less<>> {
   // (for doubles, use inf = 1/.0, div(a,b) = a/b)
   const ll inf = LLONG_MAX / 2;
   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()) {
         x->p = inf;
         return false;
      }
      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) {
      if(ismin) k = -k, m = -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) {
      auto l = *lower_bound(x);
      ll s = 1;
      if(ismin) s = -1;
      return s * (l.k * x + l.m);
   }
};
```

link-cut.hpp

md5: e9b023

```
struct Node {
   typedef Node* NP;
   NP l, r, p;
   bool rev;
   int v, mx, lz;
   Node() : l(NULL), r(NULL), p(NULL), rev(false), v(-inf),
mx(-inf), lz(-inf) {}
   void Propagate() {
      if(rev) {
         swap(l, r);
         if(l) l->rev ^= true;
         if(r) r->rev ^= true;
         rev = false;
      }
      if(l) chmax(l->lz, lz);
      if(r) chmax(r->lz, lz);
      chmax(v, lz);
      chmax(mx, lz);
      lz = -inf;
   }
   int GetMax() { return max(mx, lz); }
   int GetVert() { return max(v, lz); }
   void Update() {
      assert(lz == -inf);
      mx = v;
      if(l) { chmax(mx, l->GetMax()); }
      if(r) { chmax(mx, r->GetMax()); }
   }
   int Pos() {
      if(p && p->l == this) return -1;
      if(p && p->r == this) return 1;
      return 0;
   }
   void Prepare() {
      if(Pos()) p->Prepare();
      Propagate();
   }
   void Rotate() {
      NP q = p, c;
      if(Pos() == 1) {
         c = l;
         l = p;
         p->r = c;
      } else {
         c = r;
         r = p;
         p->l = c;
      }
      if(c) c->p = p;
      p = p->p;
      q->p = this;
      if(p && p->l == q) p->l = this;
      if(p && p->r == q) p->r = this;
      q->Update();
   }
   void Splay() {
      Prepare();
      while(Pos()) {
         int a = Pos(), b = p->Pos();
         if(b && a == b) p->Rotate();
         if(b && a != b) Rotate();
         Rotate();
      }
      Update();
   }
   void Expose() {
      for(NP x = this; x; x = x->p) x->Splay();
      for(NP x = this; x->p; x = x->p) {
         x->p->r = x;
         x->p->Update();
      }
      Splay();
   }
   void Evert() {
      Expose();
      if(l) {
         l->rev ^= true;
         l = NULL;
         Update();
      }
   }
   void Link(NP x) {
      Evert();
      p = x;
   }
   void Set(int q) {
      Expose();
      r = NULL;
      chmax(lz, q);
   }
   void Cut() {
      Expose();
      assert(l);
      l->p = NULL;
      l = NULL;
      Update();
   }
   int Get() {
      Expose();
      r = NULL;
      Update();
      return GetMax();
   }
};
Node* LCA(Node* a, Node* b) {
   a->Expose();
   b->Expose();
   if(!a->p) { return NULL; }
   Node* d = a;
   while(a->p != b) {
      if(a->Pos() == 0) { d = a->p; }
      a = a->p;
   }
   if(a == b->l) {
      return d;
   } else {
      return b;
   }
}
```

pbds.hpp

md5: a38245

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/priority_queue.hpp>
#include <ext/pb_ds/tag_and_trait.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
// using namespace __gnu_pbds;
template<typename T> using PQ = __gnu_pbds::priority_queue<T,
greater<T>, __gnu_pbds::rc_binomial_heap_tag>;
using Trie = __gnu_pbds::trie<string,
                              __gnu_pbds::null_type,

__gnu_pbds::trie_string_access_traits<>,
                              __gnu_pbds::pat_trie_tag,

__gnu_pbds::trie_prefix_search_node_update>;
// not a multiset
// find_by_order(k) -> itr of k-th(0-based) element
// order_of_key(k) -> index of lower_bound(k)
using ordered_set = tree<int, null_type, less<int>,
rb_tree_tag, tree_order_statistics_node_update>;

#include <ext/rope>
using namespace __gnu_cxx;
```

rbst.hpp

md5: b392ca

```
template<typename T, T (*f)(T, T), T (*e)()> struct RBST {
   inline int rnd() {
      static int x = 123456789;
      static int y = 362436069;
      static int z = 521288629;
      static int w = 88675123;
      int t;

      t = x ^ (x << 11);
      x = y;
      y = z;
      z = w;
      return w = (w ^ (w >> 19)) ^ (t ^ (t >> 8));
   }
   struct node {
      node *l, *r;
      int cnt;
      T x, sum;
      node() = default;
      node(T x) : x(x), sum(x), l(0), r(0) { cnt = 1; }
   };
   RBST(int n) : pool(n) {}
   int cnt(const node* t) { return t ? t->cnt : 0; }
   T sum(const node* t) { return t ? t->sum : e(); }
   node* update(node* t) {
      t->cnt = cnt(t->l) + cnt(t->r) + 1;
      t->sum = f(f(sum(t->l), t->x), sum(t->r));
      return t;
   }
   vector<node> pool;
   int ptr = 0;
   inline node* alloc(const T& v) {
      if(si(pool) == ptr) pool.resize(si(pool) * 2);
      return &(pool[ptr++] = node(v));
   }
   node* merge(node* l, node* r) {
      if(!l or !r) return l ? l : r;
      if(rnd() % (cnt(l) + cnt(r)) < cnt(l)) {
         l->r = merge(l->r, r);
         return update(l);
      }
      r->l = merge(l, r->l);
      return update(r);
   }

   pair<node*, node*> split(node* t, int k) {
      if(!t) return {t, t};
      if(k <= cnt(t->l)) {
         auto [l, r] = split(t->l, k);
         t->l = r;
         return {l, update(t)};
      }
      auto [l, r] = split(t->r, k - cnt(t->l) - 1);
      t->r = l;
      return {update(t), r};
   }

   void insert(node*& t, int k, const T& v) {
      auto [l, r] = split(t, k);
      t = merge(merge(l, alloc(v)), r);
   }
};
```

segbeats.hpp

md5: 2fbe43

```
struct Segtree_beats {
   ll op(int type, ll x, ll y) { return type ? min(x, y) :
max(x, y); }
   bool cmp(int type, ll x, ll y) { return type ? x < y : x >
y; }
   struct alignas(32) Node {
      ll sum = 0;
      ll a1[2] = {}, a2[2] = {-INFL, INFL}, ac[2] = {1, 1}, add
= 0;
   };

   vector<Node> v;
   ll n, log, e[3] = {-INFL, INFL, 0};

   Segtree_beats() {}
   Segtree_beats(int n) : Segtree_beats(vl(n)) {}
   Segtree_beats(const vl& a) {
      n = 1, log = 0;
      while(n < si(a)) n <<= 1, log++;
      v.resize(2 * n);
      rep(i, si(a)) { v[i + n].sum = v[i + n].a1[0] = v[i +
n].a1[1] = a[i]; }
      per(i, n, 1) update(i);
   }

   // 0 : add, 1 : chmin, 2 : chmax, 3 : update
   template<int cmd> void apply(int l, int r, ll x) {
      if(l == r) return;
      l += n, r += n;
      per(i, log + 1, 1) {
         if(((l >> i) << i) != l) push(l >> i);
         if(((r >> i) << i) != r) push((r - 1) >> i);
      }
      {
         int l2 = l, r2 = r;
```

```
        while(l < r) {
            if(l & 1) _apply<cmd>(l++, x);
            if(r & 1) _apply<cmd>(--r, x);
            l >>= 1;
            r >>= 1;
        }
        l = l2;
        r = r2;
    }
    rep(i, 1, log + 1) {
        if(((l >> i) << i) != l) update(l >> i);
        if(((r >> i) << i) != r) update((r - 1) >> i);
    }
}
// 0 : max, 1 : min, 2 : sum
template<int cmd> ll fold(int l, int r) {
    if(l == r) return e[cmd];
    l += n, r += n;
    per(i, log + 1, 1) {
        if(((l >> i) << i) != l) push(l >> i);
        if(((r >> i) << i) != r) push((r - 1) >> i);
    }
    ll lx = e[cmd], rx = e[cmd];
    while(l < r) {
        if(l & 1) op<cmd>(lx, v[l++]);
        if(r & 1) op<cmd>(rx, v[--r]);
        l >>= 1;
        r >>= 1;
    }
    if constexpr(cmd <= 1) lx = op(cmd, lx, rx);
    if constexpr(cmd == 2) lx += rx;
    return lx;
}

private:
void update(int k) {
    Node& p = v[k];
    Node& l = v[k * 2 + 0];
    Node& r = v[k * 2 + 1];
    p.sum = l.sum + r.sum;
    rep(t, 2) {
        if(l.a1[t] == r.a1[t]) {
            p.a1[t] = l.a1[t];
            p.a2[t] = op(t, l.a2[t], r.a2[t]);
            p.ac[t] = l.ac[t] + r.ac[t];
        } else {
            bool f = cmp(t, l.a1[t], r.a1[t]);
            p.a1[t] = f ? l.a1[t] : r.a1[t];
            p.ac[t] = f ? l.ac[t] : r.ac[t];
            p.a2[t] = op(t, f ? r.a1[t] : l.a1[t], f ? l.a2[t]
: r.a2[t]);
        }
    }
}

void push_add(int k, ll x) {
    Node& p = v[k];
    p.sum += x << (log + __builtin_clz(k) - 31);
    rep(t, 2) {
        p.a1[t] += x;
        if(p.a2[t] != e[t]) p.a2[t] += x;
    }
    p.add += x;
}

void push(int cmd, int k, ll x) {
    Node& p = v[k];
    p.sum += (x - p.a1[cmd]) * p.ac[cmd];
    if(p.a1[cmd ^ 1] == p.a1[cmd]) p.a1[cmd ^ 1] = x;
    if(p.a2[cmd ^ 1] == p.a1[cmd]) p.a2[cmd ^ 1] = x;
    p.a1[cmd] = x;
}
void push(int k) {
    Node& p = v[k];
    if(p.add) {
        rep(t, 2) push_add(k * 2 + t, p.add);
        p.add = 0;
    }
    rep(t, 2) rep(s, 2) if(cmp(t, v[k * 2 + s].a1[t],
p.a1[t])) push(t, k * 2 + s, p.a1[t]);
}

void subtree_ch(int cmd, int k, ll x) {
        if(!cmp(cmd, v[k].a1[cmd], x)) return;
        if(cmp(cmd, x, v[k].a2[cmd])) { return push(cmd, k, x); }
        push(k);
        rep(t, 2) subtree_ch(cmd, k * 2 + t, x);
        update(k);
    }

    template<int cmd> inline void _apply(int k, ll x) {
        rep(i, 2) if(cmd >> i & 1) subtree_ch(i, k, x);
        if constexpr(cmd == 0) push_add(k, x);
    }

    template<int cmd> inline void op(ll& a, const Node& b) {
        if constexpr(cmd <= 1) a = op(cmd, a, b.a1[cmd]);
        if constexpr(cmd == 2) a += b.sum;
    }
};
```

segtree-2d.hpp

md5: 1301f7

```
template<typename T, T (*op)(T, T), T (*e)()> class RangeTree {
    private:
    int n, sz;
    vector<segtree<T, op, e>> seg;
    vector<vector<pll>> yx;
    vector<pll> sorted;

    void update_(int id, ll x, ll y, T val) {
        id += n - 1;
        int yid = lb(yx[id], pll(y, x));
        seg[id].set(yid, val);
        while(id > 0) {
            id = (id - 1) / 2;
            int yid = lb(yx[id], pll(y, x));
            seg[id].set(yid, val);
        }
    }
    T query(int lxid, int rxid, ll ly, ll ry, int k, int l, int
r) {
        if(r <= lxid || rxid <= l) return e();
        if(lxid <= l && r <= rxid) {
            int lyid = lb(yx[k], pll(ly, -INFL));
            int ryid = lb(yx[k], pll(ry, -INFL));
            return (lyid >= ryid) ? e() : seg[k].prod(lyid, ryid);
        } else {
            return op(query(lxid, rxid, ly, ry, 2 * k + 1, l, (l +
r) / 2),
                      query(lxid, rxid, ly, ry, 2 * k + 2, (l + r)
/ 2, r));
        }
    }

    public:
    // 座標, 点の値
    RangeTree(vector<pll>& cand, vector<T>& val) : n(1),
sz(si(cand)), sorted(sz) {
        while(n < sz) n *= 2;
        rep(i, sz) sorted[i] = {cand[i].first, i};
        sort(all(sorted), [&](pll& a, pll& b) {
            return (a.first == b.first) ? (cand[a.second].second <
cand[b.second].second) : (a.first < b.first);
        });
        yx.resize(2 * n - 1), seg.resize(2 * n - 1);
        rep(i, sz) {
            yx[i + n - 1] = {{sorted[i].second, sorted[i].first}};
            vector<T> arg = {val[sorted[i].second]};
            seg[i + n - 1] = segtree<T, op, e>(arg);
            sorted[i].second = cand[sorted[i].second].second;
        }
        per(i, n - 1, 0) {
            yx[i].resize(si(yx[2 * i + 1]) + si(yx[2 * i + 2]));
            if(yx[i].empty()) continue;
            merge(all(yx[2 * i + 1]), all(yx[2 * i + 2]),
yx[i].begin(), [&](pll& a, pll& b) {
                return (cand[a.first].second ==
cand[b.first].second) ? (a.second < b.second)

: (cand[a.first].second < cand[b.first].second);
            });
            vector<T> arg((int)yx[i].size());
            rep(j, si(yx[i])) arg[j] = val[yx[i][j].first];
```

```
         seg[i] = segtree<T, op, e>(arg);
      }
      rep(i, 2 * n - 1) {
         for(auto& [a, b] : yx[i]) a = cand[a].second;
      }
   }
   void update(ll x, ll y, T val) {
      int id = lb(sorted, pll(x, y));
      return update_(id, x, y, val);
   }
   T query(ll lx, ll ly, ll rx, ll ry) {
      int lxid = lb(sorted, pll(lx, -INFL));
      int rxid = lb(sorted, pll(rx, -INFL));
      return (lxid >= rxid) ? e() : query(lxid, rxid, ly, ry,
0, 0, n);
   }
};
```

segtree.hpp

md5: f8e201

```
template<class S, S (*op)(S, S), S (*e)()> struct segtree {
   segtree(int n) : segtree(vector<S>(n, e())) {}
   segtree(const vector<S>& v) : n(si(v)) {
      s = bit_ceil(unsigned(n));
      log = countr_zero(unsigned(s));
      d = vector<S>(2 * s, e());
      rep(i, n) d[s + i] = v[i];
      per(i, s, 1) update(i);
   }
   void set(int p, S x) {
      d[p += s] = x;
      rep(i, 1, log + 1) update(p >> i);
   }
   S prod(int l, int r) const {
      S sml = e(), smr = e();
      l += s, r += s;
      while(l < r) {
         if(l & 1) sml = op(sml, d[l++]);
         if(r & 1) smr = op(d[--r], smr);
         l >>= 1, r >>= 1;
      }
      return op(sml, smr);
   }
   S all_prod() const { return d[1]; }
   template<typename F> int max_right(int l, F f) const {
      if(l == n) return n;
      l += s;
      S sm = e();
      do {
         while(~l & 1) l >>= 1;
         if(!f(op(sm, d[l]))) {
            while(l < s) {
               l <<= 1;
               if(f(op(sm, d[l]))) sm = op(sm, d[l++]);
            }
            return l - s;
         }
         sm = op(sm, d[l++]);
      } while((l & -l) != l);
      return n;
   }
   template<typename F> int min_left(int r, F f) const {
      if(!r) return 0;
      r += s;
      S sm = e();
      do {
         r--;
         while(r > 1 and r & 1) r >>= 1;
         if(!f(op(d[r], sm))) {
            while(r < s) {
               r = (2 * r + 1);
               if(f(op(d[r], sm))) sm = op(d[r--], sm);
            }
            return r + 1 - s;
         }
         sm = op(d[r], sm);
      } while((r & -r) != r);
      return 0;
   }

   private:
   int n, s, log;
   vector<S> d;
   void update(int k) { d[k] = op(d[k * 2], d[k * 2 + 1]); }
};
```

sparse-table-disjoint.hpp

md5: 198e80

```
template<typename T, typename F> struct sptable {
   const F f;
   vector<vector<T>> a;
   vi l;

   sptable(const vector<T>& v, F f) : f(f) {
      int m = 0;
      while((1 << m) <= si(v)) ++m;
      a.resize(m, vector<T>(si(v), T()));
      rep(i, si(v)) a[0][i] = v[i];
      rep(i, 1, m) {
         int s = 1 << i;
         for(int j = 0; j < si(v); j += s * 2) {
            int t = min(j + s, si(v));
            a[i][t - 1] = v[t - 1];
            per(k, t - 1, j) a[i][k] = f(v[k], a[i][k + 1]);
            if(si(v) <= t) break;
            a[i][t] = v[t];
            int r = min(t + s, si(v));
            rep(k, t + 1, r) a[i][k] = f(a[i][k - 1], v[k]);
         }
      }
      l.resize(1 << m);
      rep(i, 2, si(l)) l[i] = l[i >> 1] + 1;
   }

   T query(int x, int y) {
      if(x >= --y) return a[0][x];
      int p = l[x ^ y];
      return f(a[p][x], a[p][y]);
   }
};
```

swag.hpp

md5: 85c3df

```
template<typename T, typename F> struct SWAG {
   using vp = vector<pair<T, T>>;
   vp a, b;
   F f;
   T I;
   SWAG(F f, T i) : f(f), I(i) {}

   private:
   T get(vp& v) { return empty(v) ? I : v.back().second; }
   void pusha(T x) { a.eb(x, f(x, get(a))); }
   void pushb(T x) { b.eb(x, f(get(b), x)); }  // reversed!!
   void rebalance() {
      int n = si(a) + si(b);
      int s0 = n / 2 + (empty(a) ? n & 1 : 0);
      vp v{a};
      reverse(all(v));
      copy(all(b), back_inserter(v));
      a.clear(), b.clear();
      per(i, s0, 0) pusha(v[i].first);
      rep(i, s0, n) pushb(v[i].first);
   }

   public:
   T front() { return (a.empty() ? b.front() : a.back()).first;
}
   T back() { return (b.empty() ? a.front() : b.back()).first;
}
   void pop_front() {
      if(empty(a)) rebalance();
      a.pop_back();
   }
   void pop_back() {
      if(empty(b)) rebalance();
      b.pop_back();
   }
   T query() { return f(get(a), get(b)); }
};
```

wavelet_matrix.hpp

md5: dec827

```
#define U uint32_t
#define L uint64_t
```

```
struct bit_vector {
   static constexpr U w = 64;
   vector<L> block;
   vector<U> count;
   int n, zeros;

   inline U get(U i) const { return U(block[i / w] >> (i % w))
& 1; }
   inline void set(U i) { block[i / w] |= 1LL << (i % w); }

   bit_vector() {}
   bit_vector(int n) { init(n); }
   void init(int _n) {
      n = zeros = _n;
      block.resize(n / w + 1, 0);
      count.resize(si(block), 0);
   }

   void build() {
      rep(i, 1, si(block)) count[i] = count[i - 1] +
popcount(block[i - 1]);
      zeros = rank0(n);
   }

   inline U rank0(U i) const { return i - rank1(i); }
   inline U rank1(U i) const { return count[i / w] +
popcount(block[i / w] & ((1ULL << i % w) - 1)); }
};

template<typename T, const int lg = 31> struct WaveletMatrix {
   int n;
   vector<T> a;
   array<bit_vector, lg> bv;
   WaveletMatrix(const vector<T>& _a) : n(_a.size()), a(_a) {
build2(); }

   void build() {
      rep(i, lg) bv[i] = bit_vector(n);
      vector<T> cur = a, nxt(n);
      per(h, lg, 0) {
         rep(i, n) if(cur[i] >> h & 1) bv[h].set(i);
         bv[h].build();
         array<decltype(begin(nxt)), 2> it{begin(nxt),
begin(nxt) + bv[h].zeros};
         rep(i, n) * it[bv[h].get(i)]++ = cur[i];
         swap(cur, nxt);
      }
      return;
   }

   inline pair<U, U> succ0(int l, int r, int h) const { return
make_pair(bv[h].rank0(l), bv[h].rank0(r)); }

   inline pair<U, U> succ1(int l, int r, int h) const {
      U l0 = bv[h].rank0(l);
      U r0 = bv[h].rank0(r);
      U zeros = bv[h].zeros;
      return make_pair(l + zeros - l0, r + zeros - r0);
   }

   T access(U k) const {
      T ret = 0;
      per(h, lg, 0) {
         U f = bv[h].get(k);
         ret |= f ? T(1) << h : 0;
         k = f ? bv[h].rank1(k) + bv[h].zeros : bv[h].rank0(k);
      }
      return ret;
   }

   T kth_smallest(U l, U r, U k) const {
      T res = 0;
      for(int h = lg - 1; h >= 0; --h) {
         U l0 = bv[h].rank0(l), r0 = bv[h].rank0(r);
         if(k < r0 - l0) l = l0, r = r0;
         else {
            k -= r0 - l0;
            res |= (T)1 << h;
            l += bv[h].zeros - l0, r += bv[h].zeros - r0;
         }
      }
      return res;
   }
   T kth_largest(int l, int r, int k) { return kth_smallest(l,
r, r - l - k - 1); }

   int range_freq(int l, int r, T upper) {
      if(upper >= (T(1) << lg)) return r - l;
      int ret = 0;
      per(h, lg, 0) {
         bool f = (upper >> h) & 1;
         U l0 = bv[h].rank0(l), r0 = bv[h].rank0(r);
         if(f) {
            ret += r0 - l0;
            l += bv[h].zeros - l0;
            r += bv[h].zeros - r0;
         } else {
            l = l0;
            r = r0;
         }
      }
      return ret;
   }
   int range_freq(int l, int r, T lower, T upper) { return
range_freq(l, r, upper) - range_freq(l, r, lower); }

   array<vector<ll>, lg> sums;
   vector<ll> acc;
   void build2() {
      rep(i, lg) bv[i] = bit_vector(n), sums[i].assign(n + 1,
0);
      acc.resize(si(a) + 1);
      vector<T> cur = a, nxt(n);
      per(h, lg, 0) {
         rep(i, n) if((cur[i] >> h) & 1) bv[h].set(i);
         bv[h].build();
         array<decltype(begin(nxt)), 2> it{begin(nxt),
begin(nxt) + bv[h].zeros};
         rep(i, n) * it[bv[h].get(i)]++ = cur[i];
         swap(cur, nxt);
         rep(i, n) sums[h][i + 1] = sums[h][i] + cur[i];
      }
      rep(i, n) acc[i + 1] = acc[i] + a[i];
   }

   ll bottom_k_sum(int l, int r, int k) {
      ll res = 0;
      per(h, lg, 0) {
         U l0 = bv[h].rank0(l), r0 = bv[h].rank0(r);
         if(k < r0 - l0) {
            l = l0, r = r0;
         } else {
            res += sums[h][r0] - sums[h][l0];
            k -= r0 - l0;
            l += bv[h].zeros - l0;
            r += bv[h].zeros - r0;
         }
      }
      res += sums[0][l + k] - sums[0][l];
      return res;
   }

   ll top_k_sum(int l, int r, int k) { return acc[r] - acc[l] -
bottom_k_sum(l, r, r - l - k); }
};
#undef U
#undef L
```

dp

d-edge-monge.hpp

md5: 4ab7ee

```
template<class C, class T = decltype(std::declval<C>().get())>
T incremental_monge_shortest_path(const int n, C init) {
   class env {
  public:
      C mid;
      C last;
      int prev;
   };
   std::vector<env> nodes;
   {
      int n_ = n;
      int d = 0;
      while(n_ != 0) {
         n_ /= 2;
         d += 1;
      }
      nodes.assign(d, {init, init, 0});
   }
   std::vector<T> dp(n + 1, static_cast<T>(0));

   const auto f = [&](const auto& f, const int d, const int r)
-> int {
      auto& [mid, last, prev] = nodes[d];
      const int w = 1 << d;
      if((r >> d) % 2 == 1) {
         for(int i = std::max(0, r - 2 * w); i != r; i += 1) {
mid.push_back(i); }
         const int next = r + w <= n ? f(f, d + 1, r + w) : r -
w;
         int argmin = prev;
         dp[r] = dp[argmin] + mid.get();
         for(int i = prev; i != next;) {
            mid.pop_front(i);
            i += 1;
            const T t = dp[i] + mid.get();
            if(dp[r] > t) {
               dp[r] = t;
               argmin = i;
            }
         }
         prev = next;
         return argmin;
      } else {
         for(int i = std::max(0, r - 2 * w); i != r; i += 1) {
last.push_back(i); }
         for(int i = std::max(0, r - 3 * w); i != r - 2 * w; i
+= 1) { last.pop_front(i); }
         int argmin = prev;
         for(int i = r - 2 * w; i != r - w;) {
            last.pop_front(i);
            i += 1;
            const T t = dp[i] + last.get();
            if(dp[r] > t) {
               dp[r] = t;
               argmin = i;
            }
         }
         return argmin;
      }
   };

   for(int i = 1; i != n + 1; i += 1) { f(f, 0, i); }

   return dp[n];
}

namespace golden_section_search_impl {

using i64 = std::int64_t;

template<class F, class T = decltype(std::declval<F>()
(std::declval<i64>())), class Compare = std::less<T>>
std::pair<i64, T> golden_section_search(F f, i64 min, i64 max,
Compare comp = Compare()) {
   assert(min <= max);

   i64 a = min - 1, x, b;
   {
      i64 s = 1, t = 2;
      while(t < max - min + 2) { std::swap(s += t, t); }
      x = a + t - s;
      b = a + t;
   }
   T fx = f(x), fy;
   while(a + b != 2 * x) {
      const i64 y = a + b - x;
      if(max < y || comp(fx, (fy = f(y)))) {
         b = a;
         a = y;
      } else {
         a = x;
         x = y;
         fx = fy;
      }
   }
   return {x, fx};
}

}  // namespace golden_section_search_impl

using golden_section_search_impl::golden_section_search;

struct cost {
   const vector<ll>* a;
   ll lambda;
   ll cost;
   void pop_front(int l) {}
   void push_back(int r) {}
   ll get() { return lambda + c }  // 最小化なら -
};

// k : 使う辺の本数
const auto f = [&](ll l) -> ll {
   auto res = incremental_monge_shortest_path(n + 1, cost{l, 0,
0}) - l * (k + 1);
   return res;
};
// L = - max(|e|) * 3, R = max(|e|) * 3
OUT(golden_section_search(f, L, R, greater<ll>()).se);
```

mo-rollback.hpp md5: 5737bf

```
struct MoRollBack {
   using ADD = function<void(int)>;
   using REM = function<void(int)>;
   using RESET = function<void()>;
   using SNAP = function<void()>;
   using ROLLBACK = function<void()>;
   int w;
   vector<int> l, r, ord;
   MoRollBack(int n, int q) : w((int)sqrt(n)), ord(q) {
iota(all(ord), 0); }
   void add(int a, int b) { /* [l, r) */
      l.emplace_back(a);
      r.emplace_back(b);
   }
   void run(const ADD& add, const REM& rem, const RESET& reset,
const SNAP& snap, const ROLLBACK& rollback) {
      sort(begin(ord), end(ord), [&](int a, int b) {
         int ab = l[a] / w, bb = l[b] / w;
         if(ab != bb) return ab < bb;
         return r[a] < r[b];
      });
      reset();
      for(auto idx : ord) {
         if(r[idx] - l[idx] < w) {
            rep(i, l[idx], r[idx]) add(i);
            rem(idx);
            rollback();
         }
      }
      int nr = 0, lb = -1;
      for(auto idx : ord) {
         if(r[idx] - l[idx] < w) continue;
         int b = l[idx] / w;
         if(lb != b) {
            reset();
            lb = b;
            nr = (b + 1) * w;
         }
         while(nr < r[idx]) add(nr++);
         snap();
         per(j, (b + 1) * w, l[idx]) add(j);
         rem(idx);
         rollback();
      }
   }
};
```

mo.hpp md5: 6ff6db

```
struct Mo {
   int n;
   vector<pii> lr;
   Mo(int n) : n(n) {}
   void add(int l, int r) { lr.eb(l, r); }
   template<typename AL, typename AR, typename EL, typename ER,
typename O>
   void build(const AL& add_left, const AR& add_right, const
EL& erase_left, const ER& erase_right, const O& out) {
      int q = (int)lr.size();
      int bs = n / min<int>(n, sqrt(q));
      vector<int> ord(q);
      iota(all(ord), 0);
      sort(all(ord), [&](int a, int b) {
         int ab = lr[a].first / bs, bb = lr[b].first / bs;
         if(ab != bb) return ab < bb;
         return (ab & 1) ? lr[a].second > lr[b].second :
lr[a].second < lr[b].second;
      });
      int l = 0, r = 0;
      for(auto idx : ord) {
         while(l > lr[idx].first) add_left(--l);
         while(r < lr[idx].second) add_right(r++);
         while(l < lr[idx].first) erase_left(l++);
         while(r > lr[idx].second) erase_right(--r);
         out(idx);
      }
   }
   template<typename A, typename E, typename O> void
build(const A& add, const E& erase, const O& out) {
      build(add, add, erase, erase, out);
   }
};
```

monge-incremental-rowmin.hpp

md5: 2cff0f

```
// A[N + 1][N + 1]: Monge が i > j のみ存在しているとき、i (= 0,
..., N)行目の最小値を返す
// f(i, j, v) で、j 行目の最小値が求まっている v を用いて、A[i][j] に
アクセス
template<typename T, typename F> vector<T> monge_rowmin(int n,
const F& f) {
   vector<T> mi(n + 1, numeric_limits<T>::max());
   mi[0] = 0;
   vector<int> amin(n + 1);
   auto check = [&](int i, int j) {
      if(chmin(mi[i], f(i, j, mi))) { amin[i] = j; }
   };
   check(n, 0);
   auto solve = [&](auto&& self, int l, int r) {
      if(r - l == 1) return;
      int mid = l + r >> 1;
      rep(k, amin[l], amin[r] + 1) check(mid, k);
      self(self, l, mid);
      rep(k, l + 1, mid + 1) check(r, k);
      self(self, mid, r);
   };
   solve(solve, 0, n);
   return mi;
}
```

monotone-minima.hpp

md5: 187a2d

```
// monotone 行列の各行について、最小値を取る場所とその値を返す
template<typename T, typename F> vector<pair<int, T>>
monotone_minima(int h, int w, const F& f) {
   vector<pair<int, T>> dp(h, pair(-1, T()));
   auto rec = [&](auto&& rec, int u, int d, int l, int r) {
      if(u > d) return;
      int mid = u + d >> 1;
      auto& [idx, mi] = dp[mid];
      idx = l, mi = f(mid, l);
      rep(i, l + 1, r + 1) if(chmin(mi, f(mid, i))) idx = i;
      rec(rec, u, mid - 1, l, idx);
      rec(rec, mid + 1, d, idx, r);
   };
   rec(rec, 0, h - 1, 0, w - 1);
   return dp;
}
```

math

ExtGCD.hpp

md5: 88cb1c

```
// returns gcd(a, b) and assign x, y to integers
// s.t. ax + by = gcd(a, b) and |x| + |y| is minimized
ll extgcd(ll a, ll b, ll& x, ll& y) {
   // assert(a >= 0 && b >= 0);
   if(!b) return x = 1, y = 0, a;
   ll d = extgcd(b, a % b, y, x);
   y -= a / b * x;
   return d;
}
ll inv_mod(ll x, ll md) {
   ll y, z;
   extgcd(x, md, y, z);
   return (y % md + md) % md;
}
```

and-or-convolution.hpp

md5: da6157

```
// and / or convolution
template<bool isOR, typename T> void fzt(vector<T>& a, bool inv
= false) {
   int n = si(a);
   int m = __lg(n);
   rep(i, m) {
      rep(b, n) {
         if((b >> i & 1) == isOR) a[b] += a[b ^ 1 << i] * (inv
? -1 : 1);
      }
   }
}
```

binom.hpp

md5: db2821

```
constexpr int N = 1e6 + 100;
mint fact[N], ifact[N];
void pre() {
   fact[0] = 1;
   rep(i, 1, N) fact[i] = i * fact[i - 1];
   ifact[N - 1] = fact[N - 1].inv();
   per(i, N - 1, 0) ifact[i] = ifact[i + 1] * (i + 1);
}
mint C(int n, int m) { return (n < m or m < 0 ? 0 : fact[n] *
ifact[m] * ifact[n - m]); }
```

crt.hpp

md5: 0e9c10

```
// (rem, mod)
pll crt(const vl& b, const vl& c) {
   int n = si(b);
   ll r = 0, m = 1;
   rep(i, n) {
      ll g, im, x;
      g = extgcd(m, c[i], im, x);
      if((b[i] - r) % g) return {0, -1};
      ll tmp = (b[i] - r) / g * im % (c[i] / g);
      r += m * tmp;
      m *= c[i] / g;
   }
   return {(r % m + m) % m, m};
}
```

floor_sum.hpp

md5: 930ca0

```
// x_i=floor((a*i+b)/c), i=0,1,..n-1
// a,c>0, b>=0
ll floor_sum(ll n, ll a, ll b, ll c) {
   if(n == 0) return 0;
   ll res = 0;
   res += n * (n - 1) / 2 * (a / c);
   a %= c;
   res += n * (b / c);
   b %= c;
   if(a == 0) return res;
   ll top = (a * (n - 1) + b) / c;
   res += top * n;
   ll h = (b + 1 + c - 1) / c;
   if(h <= top) res -= floor_sum(top - h + 1, c, c * h - (b +
1), a) + top - h + 1;
   return res;
}
```

lagrange-hokan.hpp

md5: 22b9e3

```
template<typename T> T lagrange_polynomial(const vector<T>& y,
ll t) {
   int n = si(y) - 1;
   if(t <= n) return y[t];
   T ret(0);
   vector<T> dp(n + 1, 1), pd(n + 1, 1);
   rep(i, n) dp[i + 1] = dp[i] * (t - i);
   per(i, n + 1, 1) pd[i - 1] = pd[i] * (t - i);
   rep(i, n + 1) {
      T tmp = y[i] * dp[i] * pd[i] * ifact[i] * ifact[n - i];
      ret -= ((n - i) & 1 ? tmp : -tmp);
   }
   return ret;
}
```

matrix.hpp

md5: 270f1c

```
template<typename T> struct M {
   vector<vector<T>> a;
   int n, m;
   M(int n, int m) : n(n), m(m), a(n, vector<T>(m)) {}
   M(int n = 0) : M<T>(n, n) {}
   vector<T>& operator[](int k) { return a[k]; }
   const vector<T>& operator[](int k) const { return a[k]; }
   static M I(int n) {
      M mat(n);
      rep(i, n) mat[i][i] = 1;
      return mat;
   }
   M& operator+=(const M& b) {
      rep(i, n) rep(j, m)(*this)[i][j] += b[i][j];
      return *this;
   }
   M& operator-=(const M& b) {
      rep(i, n) rep(j, m)(*this)[i][j] -= b[i][j];
      return *this;
   }
   M& operator*=(const M& b) {
      int l = b.m;
      vector c(n, vector<T>(l));
      rep(i, n) rep(j, m) rep(k, l) c[i][k] += (*this)[i][j] *
b[j][k];
      a.swap(c);
      return *this;
   }
   M& operator^=(ll k) {
      M b = M::I(n);
      while(k) {
         if(k & 1) b *= *this;
         *this *= *this;
         k >>= 1;
      }
      a.swap(b.a);
      return *this;
   }
   M operator+(const M& b) const { return (M(*this) += b); }
   M operator-(const M& b) const { return (M(*this) -= b); }
   M operator*(const M& b) const { return (M(*this) *= b); }
   M operator^(const M& b) const { return (M(*this) ^= b); }
};
template<typename T> pair<int, T> GaussElimination(M<T>& a,
bool LE = false) {
   int n = a.n, m = a.m;
   int rank = 0, je = LE ? m - 1 : m;
   mint det = 1;
   rep(j, je) {
      int idx = -1;
      rep(i, rank, n) {
         if(a[i][j].x) {
            idx = i;
            break;
         }
      }
      if(idx == -1) {
         det = 0;
         continue;
      }
      if(rank != idx) {
         det = -det;
         swap(a[rank], a[idx]);
      }
      det *= a[rank][j];
      if(LE && a[rank][j].x != 1) {
         mint coeff = a[rank][j].inv();
         rep(k, j, m) a[rank][k] *= coeff;
      }
      int is = LE ? 0 : rank + 1;
      rep(i, is, n) {
         if(i == rank) continue;
         if(a[i][j].x) {
            mint coeff = a[i][j] / a[rank][j];
            rep(k, j, m) a[i][k] -= a[rank][k] * coeff;
         }
      }
      rank++;
   }
   return make_pair(rank, det);
}

template<typename T> vector<vector<T>> LinearEquation(M<T> a,
vector<T> b) {
   int n = a.n, m = a.m;
   rep(i, n) a[i].eb(b[i]);
   a.m++;
   auto p = GaussElimination(a, true);
   int rank = p.first;
   rep(i, rank, n) {
      if(a[i][m].x != 0) return {};
   }
   vector<vector<T>> res(1, vector<T>(m));
   vi piv(m, -1);
   int j = 0;
   rep(i, rank) {
      while(a[i][j].x == 0) ++j;
      res[0][j] = a[i][m], piv[j] = i;
   }
   rep(j, m) {
      if(piv[j] == -1) {
         vector<T> x(m);
         x[j] = 1;
         rep(k, j) {
            if(piv[k] != -1) x[k] = -a[piv[k]][j];
         }
         res.eb(x);
      }
   }
   return res;
}
```

prime.hpp

md5: 94a4a8

```
template<class T, class U> T pow_mod(T x, U n, T md) {
   T r = 1 % md;
   x %= md;
   while(n) {
      if(n & 1) r = (r * x) % md;
      x = (x * x) % md;
      n >>= 1;
   }
   return r;
}
bool is_prime(ll n) {
   if(n <= 1) return false;
   if(n == 2) return true;
   if(n % 2 == 0) return false;
   ll d = n - 1;
   while(d % 2 == 0) d /= 2;
   for(ll a : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37}) {
      if(n <= a) break;
      ll t = d;
      ll y = pow_mod<i128>(a, t, n);  // over
      while(t != n - 1 && y != 1 && y != n - 1) {
         y = i128(y) * y % n;  // flow
         t <<= 1;
      }
      if(y != n - 1 && t % 2 == 0) { return false; }
   }
   return true;
}
ll pollard_single(ll n) {
   ll R;
   auto f = [&](ll x) { return (i128(x) * x + R) % n; };
   if(is_prime(n)) return n;
   if(n % 2 == 0) return 2;
   ll st = 0;
   while(true) {
      R = rnd(1, n);
      st++;
      ll x = st, y = f(x);
      while(true) {
         ll p = gcd((y - x + n), n);
         if(p == 0 || p == n) break;
         if(p != 1) return p;
         x = f(x);
         y = f(f(y));
      }
   }
}
vl factor(ll n) {
   if(n == 1) return {};
   ll x = pollard_single(n);
   if(x == n) return {x};
   vl l = factor(x), r = factor(n / x);
   return l.insert(end(l), all(r)), l;
}
```

primitive-root.hpp

md5: 4088f5

```
ll primitive_root(ll p) {
   auto v = factor(p - 1);
   sort(all(v)), v.erase(unique(all(v)), end(v));
   while(true) {
      ll g = rnd(1, p);
      bool ok = true;
      for(auto d : v) {
         ll f = (p - 1) / d;
         if(pow_mod<i128>(g, f, p) == 1) {
            ok = false;
            break;
         }
      }
      if(ok) return g;
   }
}
```

xor-convolution.hpp

md5: f5168d

```
template<typename T> void fwt(vector<T>& f, bool inv = false) {
   int n = si(f), m = __lg(n);
   rep(i, m) {
      rep(b, n) {
         if(~b >> i & 1) {
            T x = f[b], y = f[b ^ 1 << i];
            f[b] = x + y, f[b ^ 1 << i] = x - y;
         }
      }
   }
   if(inv) {
      T iz = T(1) / T(si(f));
      fore(e, f) e *= iz;
   }
}
```

graph

bcc.hpp

md5: 3df588

```
template<typename G> struct BCC : LL<G> {
   vi used;
   vector<vector<pii>> bc;
   vector<pii> tmp;
   using L = LL<G>;
   using L::g;
   using L::low;
   using L::ord;

   BCC(G g) : L(g) { build(); }

   void build() {
      used.assign(si(g), 0);
      rep(i, si(used)) if(!used[i]) dfs(i, -1);
   }

   void dfs(int x, int p) {
      used[x] = true;
      fore(e, g[x]) {
         if(e == p) continue;
         if(!used[e] || ord[e] < ord[x]) tmp.eb(minmax(x, e));
         if(!used[e]) {
            dfs(e, x);
            if(low[e] >= ord[x]) {
               bc.eb();
               while(true) {
                  auto p = tmp.back();
                  bc.back().eb(p);
                  tmp.pop_back();
                  if(p.first == min(x, e) and p.second ==
max(x, e)) break;
               }
            }
         }
      }
   }
};
```

eulerian-trail.hpp

md5: 5d421d

```
struct edge {
   int x, y, idx;
};
vector<edge> eulerian_path(vector<edge> es, int s, bool
directed = false) {
   if(es.empty()) return {};
   int n = 0;
   fore(e, es) chmax(n, max(e.x, e.y) + 1);
   vector<vector<pair<edge, int>>> g(n);
   for(auto& e : es) {
      int p = si(g[e.y]);
      g[e.x].emplace_back(e, p);
      if(!directed) {
         int q = si(g[e.x]) - 1;
         swap(e.x, e.y);
         g[e.x].emplace_back(e, q);
      }
   }
   vector<edge> ord;
   stack<pair<int, edge>> st;
   st.emplace(s, edge{-1, -1, -1});
   while(st.size()) {
      int x = st.top().first;
      if(empty(g[x])) {
         ord.eb(st.top().second);
         st.pop();
      } else {
         auto e = g[x].back();
         g[x].pop_back();
         if(e.second == -1) continue;
         if(!directed) g[e.first.y][e.second].second = -1;
         st.emplace(e.first.y, e.first);
      }
   }
   ord.pop_back();
   reverse(begin(ord), end(ord));
   if(si(ord) != si(es)) return {};
   return ord;
}
```

lowlink.hpp

md5: e3987c

```
template<typename G> struct LL {
   int n;
   const G g;
   vi ord, low, arti;
   vector<pii> bridge;

   LL(G g) : n(si(g)), g(g), ord(si(g), -1), low(si(g), -1) {
      int k = 0;
      rep(i, n) {
         if(ord[i] == -1) k = dfs(i, k, -1);
      }
   int dfs(int x, int k, int p) {
      low[x] = (ord[x] = k++);
      int cnt = 0;
      bool is_arti = false, second = false;
      fore(e, g[x]) {
         if(ord[e] == -1) {
            cnt++;
            k = dfs(e, k, x);
            chmin(low[x], low[e]);
            is_arti |= (p != -1) && (low[e] >= ord[x]);
            if(ord[x] < low[e]) bridge.eb(minmax(x, e));
         } else if(e != p or second) {
            chmin(low[x], ord[e]);
         } else {
            second = true;
         }
      }
      is_arti |= p == -1 && cnt > 1;
      if(is_arti) arti.eb(x);
      return k;
   }
};
```

max_matching.hpp

md5: 2ece25

```
struct Matching {
   int n;
   vector<vi> g;
   vi mt;
   vi is_ev, gr_buf;
   vector<pii> nx;
   int st;
   int group(int x) {
      if(gr_buf[x] == -1 || is_ev[gr_buf[x]] != st) return
gr_buf[x];
      return gr_buf[x] = group(gr_buf[x]);
   }
   void match(int p, int b) {
      int d = mt[p];
      mt[p] = b;
      if(d == -1 || mt[d] != p) return;
      if(nx[p].second == -1) {
         mt[d] = nx[p].first;
         match(nx[p].first, d);
      } else {
         match(nx[p].first, nx[p].second);
         match(nx[p].second, nx[p].first);
      }
   }
   bool arg() {
      is_ev[st] = st;
      gr_buf[st] = -1;
      nx[st] = pii(-1, -1);
      queue<int> q;
      q.push(st);
      while(q.size()) {
         int a = q.front();
         q.pop();
         for(auto b : g[a]) {
            if(b == st) continue;
            if(mt[b] == -1) {
               mt[b] = a;
               match(a, b);
               return true;
            }
            if(is_ev[b] == st) {
               int x = group(a), y = group(b);
               if(x == y) continue;
               int z = -1;
               while(x != -1 || y != -1) {
                  if(y != -1) swap(x, y);
                  if(nx[x] == pii(a, b)) {
                     z = x;
                     break;
                  }
                  nx[x] = pii(a, b);
                  x = group(nx[mt[x]].first);
               }
               for(int v : {group(a), group(b)}) {
                  while(v != z) {
                     q.push(v);
                     is_ev[v] = st;
                     gr_buf[v] = z;
                     v = group(nx[mt[v]].first);
                  }
               }
            } else if(is_ev[mt[b]] != st) {
               is_ev[mt[b]] = st;
               nx[b] = pii(-1, -1);
               nx[mt[b]] = pii(a, -1);
               gr_buf[mt[b]] = b;
               q.push(mt[b]);
            }
         }
      }
      return false;
   }
   Matching(const vector<vi>& _g) : n(int(_g.size())), g(_g),
mt(n, -1), is_ev(n, -1), gr_buf(n), nx(n) {
      for(st = 0; st < n; st++)
         if(mt[st] == -1) arg();
   }
   vector<pii> max_match() {
      vector<pii> res;
      rep(i, n) if(i < mt[i]) res.eb(i, mt[i]);
      return res;
   }
};
```

maximum-independent-set.hpp

md5: ac1384

```
unsigned ll maximum_independent_set(vector<vi> g) {
   using U = unsigned long long;
   int n = si(g);
   vector<U> nbd(n);
   rep(i, n) fore(e, g[i]) nbd[i] |= 1ULL << e;
   int best = 0;
   U res = 0;
   auto dfs = [&](auto&& dfs, U now, U rest) -> void {
      pii p(-1, -1);
      while(true) {
         bool upd = 0;
         rep(v, n) {
            if(rest >> v & 1) {
               int d = popcount(nbd[v] & rest);
               if(chmax(p.second, d)) p.first = v;
               if(d <= 1) rest ^= 1ULL << v, rest &= ~nbd[v],
now |= 1ULL << v, upd = 1;
            }
         }
         if(!upd) break;
         p = {-1, -1};
      }
      int a = popcount(now), b = popcount(rest);
      if(chmax(best, a)) res = now;
      if(!b or a + b <= best) return;
      int v = p.first;
      rest &= ~(1ULL << v);
      if(p.second >= 3) dfs(dfs, now, rest);
      now |= 1ULL << v;
      dfs(dfs, now, rest & ~(nbd[v]));
   };
   U now = 0, rest = (1ULL << n) - 1;
   dfs(dfs, now, rest);
   return res;
}
```

scc.hpp

md5: d3e879

```
template<typename G> struct SCC {
   G g;
   vector<vi> rg;
   vi comp, ord, used;
   int num;  // 連結成分の数

   SCC(G g) : g(g), rg(si(g)), comp(si(g), -1), used(si(g)) {
      rep(i, si(g)) fore(e, g[i]) rg[e].eb(i);
      build();
   };
   int operator[](int k) { return comp[k]; }
   void dfs(int x) {
      if(used[x]) return;
      used[x] = true;
      fore(e, g[x]) if(!used[e]) dfs(e);
      ord.eb(x);
   }
   void rdfs(int x, int cnt) {
      if(comp[x] != -1) return;
      comp[x] = cnt;
      fore(e, rg[x]) if(comp[e] == -1) rdfs(e, cnt);
   }
   void build() {
      rep(i, g.size()) dfs(i);
      reverse(all(ord));
      num = 0;
      fore(i, ord) if(comp[i] == -1) { rdfs(i, num), num++; }
   }
};
```

tecc.hpp

md5: 17c69f

```
template<typename G> struct TCC : LL<G> {
   using L = LL<G>;
   using L::bridge;
   using L::g;
   using L::low;
   using L::ord;
   vi cmp;
   vector<vi> tree, group;
   void build() {
      cmp.assign(si(g), -1);
      int k = 0;
      rep(i, si(cmp)) if(cmp[i] == -1) dfs(i, -1, k);
      group.resize(k);
      rep(i, si(g)) group[cmp[i]].eb(i);
      tree.resize(k);
      for(auto [a, b] : bridge) {
         tree[cmp[a]].eb(cmp[b]);
         tree[cmp[b]].eb(cmp[a]);
      }
   }
   TCC(const G& g) : L(g) { build(); }
   void dfs(int x, int p, int& k) {
      if(p >= 0 and ord[p] >= low[x]) cmp[x] = cmp[p];
      else cmp[x] = k++;
      fore(e, g[x]) if(cmp[e] == -1) dfs(e, x, k);
   }
};
```

modint

BarrettReduction.hpp

md5: 651912

```
using U = uint64_t;
struct Barret {
   U m, im;
   Barret(U mod) : m(mod), im(-1ULL / m + 1) {}
   U mul(U a, U b) const {
      a *= b;
      U x = ((__uint128_t)a * im) >> 64;
      a -= x * m;
      if((ll)a < 0) a += m;
      return a;
   }
};

constexpr ll mod = 998244353;
static Barret b(mod);
struct mint {
   int x;
   mint(ll x_ = 0) : x((x_ % mod) + mod) {
      if(x >= mod) x -= mod;
   }
   mint& s(uint xx) { return x = xx < mod ? xx : xx - mod,
*this; }
   mint operator-() { return mint(-x); }
   mint& operator+=(const mint& r) { return s(x + r.x); }
   mint& operator-=(const mint& r) { return s(x + mod - r.x); }
   mint& operator*=(const mint& r) { return x = b.mul(x, r.x),
*this; }
   mint& operator/=(const mint& r) { return *this *= r.inv(); }
   friend mint operator+(mint l, mint r) { return l += r; }
   friend mint operator-(mint l, mint r) { return l -= r; }
   friend mint operator*(mint l, mint r) { return l *= r; }
   friend mint operator/(mint l, mint r) { return l /= r; }
   mint inv() const { return pow(mod - 2); }
   mint pow(ll b) const {
      mint a = *this, c = 1;
      while(b) {
         if(b & 1) c *= a;
         a *= a;
         b >>= 1;
      }
      return c;
   }
};
using vm = vector<mint>;
```

modint.hpp

md5: 3db9f2

```
constexpr int mod = 998244353;
struct mint {
   int x;
   mint(ll x_ = 0) : x(x_ % mod) {
      if(x < 0) x += mod;
   }
   mint operator-() {
      auto res = *this;
      res.x = (x ? mod - x : 0);
      return res;
   }
   mint& operator+=(mint r) {
      if((x += r.x) >= mod) x -= mod;
      return *this;
   }
   mint& operator-=(mint r) {
      if((x -= r.x) < 0) x += mod;
      return *this;
   }
   mint& operator*=(mint r) {
      x = 1LL * x * r.x % mod;
      return *this;
   }
   mint& operator/=(mint r) { return *this *= r.inv(); }
   friend mint operator+(mint a, mint b) { return a += b; }
   friend mint operator-(mint a, mint b) { return a -= b; }
   friend mint operator*(mint a, mint b) { return a *= b; }
   friend mint operator/(mint a, mint b) { return a /= b; }
   mint inv() const { return pow(mod - 2); }
   mint pow(ll b) const {
      mint a = *this, c = 1;
      while(b) {
         if(b & 1) c *= a;
         a *= a;
         b >>= 1;
      }
      return c;
   }
};
using vm = vector<mint>;
```

FPS

FFT.hpp

md5: f769b5

```
mint g = 3;  // 原始根
void fft(vm& a, bool inv = false) {
   int n = si(a), s = __lg(n);
   static vm z, iz;
   while(si(z) <= s) {
      z.eb(g.pow(mint(-1).x / (1 << si(z))));
      iz.eb(z.back().inv());
   }
   vm b(n);
   rep(i, 1, s + 1) {
      int w = 1 << s - i;
      mint base = inv ? iz[i] : z[i], now = 1;
      for(int y = 0; y < n / 2; y += w) {
         rep(x, w) {
            auto l = a[y << 1 | x], r = now * a[y << 1 | x |
w];
            b[y | x] = l + r, b[y | x | n >> 1] = l - r;
         }
         now *= base;
      }
      swap(a, b);
   }
```

```
vm mul(vm a, vm b) {
   int n = si(a), m = si(b);
   if(!n or !m) return {};
   if(min(n, m) <= 30) {
      vm ans(n + m - 1);
      rep(i, n) rep(j, m) ans[i + j] += a[i] * b[j];
      return ans;
   }
   int N = n + m - 1;
   int z = bit_ceil(unsigned(N));
   a.resize(z), b.resize(z);
   fft(a), fft(b);
   rep(i, z) a[i] *= b[i];
   fft(a, true);
   a.resize(n + m - 1);
   mint iz = mint(z).inv();
   fore(e, a) e *= iz;
   return a;
}
```

linear-recurrence.hpp

md5: 7ef16a

```
// [x ^ k] p / q
mint LinearRecurrence(ll k, fps q, fps p) {
   q.shrink();
   mint ret = 0;
   if(si(p) >= si(q)) {
      auto r = p / q;
      p -= r * q;
      p.shrink();
      if(k < r.size()) ret += r[k];
   }
   if(p.size() == 0) return ret;
   p.resize(q.size() - 1);
   while(k) {
      auto q2 = q;
      for(int i = 1; i < q2.size(); i += 2) q2[i] = -q2[i];
      auto s = p * q2, t = q * q2;
      for(int i = (k & 1); i < s.size(); i += 2) p[i >> 1] =
s[i];
      for(int i = 0; i < t.size(); i += 2) q[i >> 1] = t[i];
      k >>= 1;
   }
   return ret + p[0];
}
// a * q = 0
mint kitamasa(ll n, fps q, fps a) {
   if(n < si(a)) return a[n];
   auto p = a.pre(si(q) - 1) * q;
   p.resize(si(q) - 1);
   return LinearRecurrence(n, q, p);
}
```

poly.hpp

md5: 8da6ee

```
struct fps {
   vm v;
   fps(const vm& v = {}) : v(v) {}
   fps(int n) : v(n) {}
   void shrink() {
      while(v.size() && !v.back().x) v.pop_back();
   }
   void resize(int n) { v.resize(n); }
   int size() const { return int(v.size()); }
   mint freq(int p) const { return (p < size()) ? v[p] : 0; }
   mint& operator[](int k) { return v[k]; }
   void emplace_back(mint x) { v.eb(x); }
   fps pre(int le) const { return {{v.begin(), v.begin() +
min(size(), le)}}; }
   fps operator-() const {
      vm res{v};
      fore(e, res) e = -e;
      return res;
   }
   fps operator+(const fps& r) const {
      auto n = max(size(), r.size());
      vm res(n);
      rep(i, n) res[i] = freq(i) + r.freq(i);
      return res;
   }
   fps operator-(const fps& r) const { return (*this) + (-r); }
   fps operator*(const fps& r) const { return {mul(v, r.v)}; }
   fps operator*(const mint& r) const {
      int n = size();
      vm res(n);
      for(int i = 0; i < n; i++) res[i] = v[i] * r;
      return res;
   }
   fps operator/(const mint& r) const { return *this * r.inv();
}
   fps operator/(const fps& r) const {
      if(size() < r.size()) return {{}};
      int n = size() - r.size() + 1;
      return (rev().pre(n) * r.rev().inv(n)).pre(n).rev();
   }
   fps operator%(const fps& r) const { return *this - *this / r
* r; }
   fps operator<<(int s) const {
      vm res(size() + s);
      rep(i, size()) res[i + s] = v[i];
      return res;
   }
   fps operator>>(int s) const {
      if(size() <= s) return fps();
      vm res(size() - s);
      rep(i, size() - s) res[i] = v[i + s];
      return res;
   }
   fps& operator+=(const fps& r) { return *this = *this + r; }
   fps& operator-=(const fps& r) { return *this = *this - r; }
   fps& operator*=(const fps& r) { return *this = *this * r; }
   fps& operator*=(const mint& r) { return *this = *this * r; }
   fps& operator/=(const fps& r) { return *this = *this / r; }
   fps& operator/=(const mint& r) { return *this = *this / r; }
   fps& operator%=(const fps& r) { return *this = *this % r; }
   fps& operator<<=(int n) { return *this = *this << n; }
   fps& operator>>=(int n) { return *this = *this >> n; }
   fps rev(int n = -1) const {
      vm res = v;
      if(n != -1) res.resize(n);
      reverse(res.begin(), res.end());
      return res;
   }
   fps diff() const {
      vm res(max(0, size() - 1));
      rep(i, 1, size()) res[i - 1] = freq(i) * i;
      return res;
   }
   fps integ() const {
      vm res(size() + 1);
      rep(i, size()) res[i + 1] = freq(i) / (i + 1);
      return res;
   }
   // f * f.inv() = 1 + g(x)x^m
   fps inv(int m) const {
      fps res = fps(vm{mint(1) / freq(0)});
      for(int i = 1; i < m; i *= 2) { res = (res * mint(2) -
res * res * pre(2 * i)).pre(2 * i); }
      return res.pre(m);
   }
   fps exp(int n) const {
      assert(freq(0).x == 0);
      fps g = fps(vm{1});
      for(int i = 1; i < n; i *= 2) { g = (g * (pre(i * 2) +
fps(vm{1}) - g.log(i * 2))).pre(i * 2); }
      return g.pre(n);
   }
   fps log(int n) const {
      assert(freq(0).x == 1);
      auto f = pre(n);
      return (f.diff() * f.inv(n - 1)).pre(n - 1).integ();
   }
   fps sqrt(int n) const {
      assert(freq(0).x == 1);
      fps f = pre(n + 1);
      fps g({1});
      for(int i = 1; i < n; i *= 2) { g = (g + f.pre(2 * i) *
g.inv(2 * i)) * mint((mod + 1) / 2); }
      return g.pre(n + 1);
   }
   fps pow(ll k, ll n) {
      if(k == 0) {
         fps res(n);
         res[0] = 1;
         return res;
      }
      rep(i, size()) {
         if((*this)[i].x) {
            mint rev = mint(1) / (*this)[i];
            fps ret = (((*this * rev) >> i).log(n) *
mint(k)).exp(n);
            ret *= (*this)[i].pow(k);
            ret = (ret << (i * k)).pre(n);
            if(ret.size() < n) ret.resize(n);
            return ret;
         }
         if(i128(i + 1) * k >= n) return fps(n);
      }
      return fps(n);
   }
   fps pow_mod(ll n, const fps& mod) {
      fps x = *this, r = {{1}};
      while(n) {
         if(n & 1) r = r * x % mod;
         x = x * x % mod;
         n >>= 1;
      }
      return r;
   }
};
```

relaxed-convolution.hpp

md5: f1c765

```
struct relaxed_multiplication {
   vector<mint> f, g, h;

   // fg_prefix_ntts[d] = (NTTs of first 2^d terms of f and g)
   vector<pair<vector<mint>, vector<mint>>> fg_prefix_ntts;

   const auto& get_fg_prefix_ntt(int d) {
      while(int(fg_prefix_ntts.size()) <= d) {
         int fftlen = 1 << fg_prefix_ntts.size();
         vector<mint> vf(f.begin(), f.begin() + fftlen);
         vector<mint> vg(g.begin(), g.begin() + fftlen);
         ntt(vf, false), ntt(vg, false);
         fg_prefix_ntts.emplace_back(vf, vg);
      }
      return fg_prefix_ntts[d];
   }

   relaxed_multiplication() {}

   mint add(const mint& f_i, const mint& g_i) {
      f.push_back(f_i), g.push_back(g_i);
      const int n = f.size(), d = __builtin_ctz(n), D = 1 << d;

      if(int gsz = n - 1 + D; h.size() < gsz) h.resize(gsz);

      if(n == D) {
         // Convolve f[0, D) * g[0, D) -> h[D - 1, D * 2 - 1)

         const auto& [nttf, nttg] = get_fg_prefix_ntt(d);
         vector<mint> tmp(nttf.size());
         for(int i = 0; i < nttf.size(); ++i) tmp[i] = nttf[i]
* nttg[i];
         ntt(tmp, true);

         for(int i = 0; i < n - 1; ++i) h[n + i] += tmp[i] -
h[i];  // 回り込みを削除
         h[n - 1] += tmp[n - 1];
      } else {
         // Convolve f[0, 2 * D) * g[n - D, n) -> h[n - 1, n -
1 + D)

         if(d <= 4) {  // Bruteforce threshold
            for(int i = n - D; i < n; ++i) {
               for(int k = n - 1; k < n - 1 + D; ++k) { h[k] +=
f[i] * g[k - i] + f[k - i] * g[i]; }
            }
         } else {
            vector<mint> tmpf{f.end() - D, f.end()},
tmpg{g.end() - D, g.end()};
            tmpf.resize(D * 2), tmpg.resize(D * 2);
            ntt(tmpf, false), ntt(tmpg, false);

            const auto& [nttf, nttg] = get_fg_prefix_ntt(d +
1);
            for(int i = 0; i < tmpf.size(); ++i) { tmpf[i] =
tmpf[i] * nttg[i] + tmpg[i] * nttf[i]; }
            ntt(tmpf, true);
            for(int i = 0; i < D; ++i) h[n - 1 + i] += tmpf[D -
1 + i];
         }
      }

      return h[n - 1];
   }
};
```

tree

block-cut-tree.hpp

md5: bf0113

```
struct extended_block_cut_tree {
   int N, cnt;
   vector<vector<int>> G;
   extended_block_cut_tree(vector<vector<int>>& E) {
      N = E.size();
      vector<int> next(N, -1);
      vector<int> d(N, -1);
      vector<int> imos(N, 0);
      for(int i = 0; i < N; i++) {
         if(d[i] == -1) {
            d[i] = 0;
            dfs1(E, next, d, imos, i);
         }
      }
      cnt = 0;
      G.resize(N + 1);
      vector<bool> used(N, false);
      for(int i = 0; i < N; i++) {
         if(d[i] == 0) { dfs2(E, d, imos, used, cnt, i); }
         if(E[i].empty()) {
            G[i].push_back(N + cnt);
            G[N + cnt].push_back(i);
            cnt++;
            G.push_back({});
         }
      }
      G.pop_back();
   }
   void dfs1(vector<vector<int>>& E, vector<int>& next,
vector<int>& d, vector<int>& imos, int v) {
      for(int w : E[v]) {
         if(d[w] == -1) {
            d[w] = d[v] + 1;
            next[v] = w;
            dfs1(E, next, d, imos, w);
            imos[v] += imos[w];
         } else if(d[w] < d[v] - 1) {
            imos[v]++;
            imos[next[w]]--;
         }
      }
   }
   void dfs2(vector<vector<int>>& E, vector<int>& d,
vector<int>& imos, vector<bool>& used, int b, int v) {
      used[v] = true;
      bool ok = false;
      for(int w : E[v]) {
         if(d[w] == d[v] + 1 && !used[w]) {
            if(imos[w] > 0) {
               if(!ok) {
                  ok = true;
                  G[v].push_back(N + b);
                  G[N + b].push_back(v);
               }
               dfs2(E, d, imos, used, b, w);
            } else {
               G[v].push_back(N + cnt);
               G[N + cnt].push_back(v);
               cnt++;
               G.push_back({});
               dfs2(E, d, imos, used, cnt - 1, w);
            }
         }
      }
      if(!ok && d[v] > 0) {
         G[v].push_back(N + b);
```

```
        G[N + b].push_back(v);
     }
  }
  int size() { return G.size(); }
  vector<int>& operator[](int v) { return G[v]; }
};
```

hld.hpp

md5: fa40a1

```
template<typename G> struct HLD {
   int n;
   G& g;
   vector<int> sub, in, out, head, rev, par, d;
   HLD(G& g) : n(si(g)), g(g), sub(n), in(n), out(n), head(n),
rev(n), par(n), d(n) {}
   void dfs1(int x, int p) {
      par[x] = p;
      sub[x] = 1;
      if(g[x].size() and g[x][0] == p) swap(g[x][0],
g[x].back());
      fore(e, g[x]) {
         if(e == p) continue;
         d[e] = d[x] + 1;
         dfs1(e, x);
         sub[x] += sub[e];
         if(sub[g[x][0]] < sub[e]) swap(g[x][0], e);
      }
   }
   void dfs2(int x, int p, int& t) {
      in[x] = t++;
      rev[in[x]] = x;
      fore(e, g[x]) {
         if(e == p) continue;
         head[e] = (g[x][0] == e ? head[x] : e);
         dfs2(e, x, t);
      }
      out[x] = t;
   }
   void build() {
      int t = 0;
      head[0] = 0;
      dfs1(0, -1);
      dfs2(0, -1, t);
   }
   int la(int v, int k) {
      while(1) {
         int u = head[v];
         if(in[v] - k >= in[u]) return rev[in[v] - k];
         k -= in[v] - in[u] + 1;
         v = par[u];
      }
   }
   int lca(int u, int v) {
      for(;; v = par[head[v]]) {
         if(in[u] > in[v]) swap(u, v);
         if(head[u] == head[v]) return u;
      }
   }
   template<typename T, typename Q, typename F>
   T query(int u, int v, const T& e, const Q& q, const F& f,
bool edge = false) {
      T l = e, r = e;
      for(;; v = par[head[v]]) {
         if(in[u] > in[v]) swap(u, v), swap(l, r);
         if(head[u] == head[v]) break;
         l = f(q(in[head[v]], in[v] + 1), l);
      }
      return f(f(q(in[u] + edge, in[v] + 1), l), r);
   }
   int dist(int u, int v) { return d[u] + d[v] - 2 * d[lca(u,
v)]; }
   int jump(int s, int t, int i) {
      if(!i) return s;
      int l = lca(s, t);
      int dst = d[s] + d[t] - d[l] * 2;
      if(dst < i) return -1;
      if(d[s] - d[l] >= i) return la(s, i);
      i -= d[s] - d[l];
      return la(t, d[t] - d[l] - i);
   }
};
```

flow

bipartite-matching.hpp

md5: 2ffb05

```
struct Bimatch {
   vector<vi> g;
   vi d, mc, used, vv;
   Bimatch(int n, int m) : g(n), mc(m, -1), used(n) {}
   void add(int u, int v) { g[u].eb(v); }
   void bfs() {
      d.assign(si(g), -1);
      queue<int> q;
      rep(i, si(g)) {
         if(!used[i]) {
            q.emplace(i);
            d[i] = 0;
         }
      }
      while(!q.empty()) {
         int x = q.front();
         q.pop();
         fore(e, g[x]) {
            int c = mc[e];
            if(c >= 0 and d[c] == -1) {
               d[c] = d[x] + 1;
               q.emplace(c);
            }
         }
      }
   }

   bool dfs(int x) {
      vv[x] = true;
      fore(e, g[x]) {
         int c = mc[e];
         if(c < 0 or (!vv[c] and d[c] == d[x] + 1 and dfs(c)))
{
            mc[e] = x;
            used[x] = true;
            return true;
         }
      }
      return false;
   }
   int match() {
      int ret = 0;
      while(true) {
         bfs();
         vv.assign(si(g), false);
         int f = 0;
         rep(i, si(g)) if(!used[i] and dfs(i)) f++;
         if(!f) return ret;
         ret += f;
      }
   }
};
```

flow.hpp

md5: e99393

```
template<typename T> struct Dinic {
   const T INF;

   struct edge {
      int to;
      T cap;
      int rev;
      bool isrev;
      int idx;
   };

   vector<vector<edge>> g;
   vector<int> c, iter;
   Dinic(int V) : INF(numeric_limits<T>::max()), g(V) {}
   void add_edge(int from, int to, T cap, int idx = -1) {
      g[from].emplace_back((edge){to, cap, si(g[to]), false,
idx});
      g[to].emplace_back((edge){from, 0, si(g[from]) - 1, true,
idx});
   }

   bool bfs(int s, int t) {
      c.assign(si(g), -1);
      queue<int> q;
      c[s] = 0;
      q.push(s);
      while(!q.empty() && c[t] == -1) {
         int x = q.front();
         q.pop();
         fore(e, g[x]) {
            if(e.cap > 0 && c[e.to] == -1) {
               c[e.to] = c[x] + 1;
               q.push(e.to);
            }
         }
      }
      return c[t] != -1;
   }

   T dfs(int x, int t, T flow) {
      if(x == t) return flow;
      for(int& i = iter[x]; i < si(g[x]); i++) {
         edge& e = g[x][i];
         if(e.cap > 0 && c[x] < c[e.to]) {
            T d = dfs(e.to, t, min(flow, e.cap));
            if(d > 0) {
               e.cap -= d;
               g[e.to][e.rev].cap += d;
               return d;
            }
         }
      }
      return 0;
   }

   T max_flow(int s, int t) {
      T flow = 0;
      while(bfs(s, t)) {
         iter.assign(si(g), 0);
         T f = 0;
         while((f = dfs(s, t, INF)) > 0) flow += f;
      }
      return flow;
   }

   // void output() {
   //     for(int i = 0; i < g.size(); i++) {
   //         for(auto &e : g[i]) {
   //             if(e.isrev) continue;
   //             auto &rev_e = g[e.to][e.rev];
   //             cout << i << "->" << e.to << " (flow: " <<
rev_e.cap << "/" << e.cap + rev_e.cap << ")" << endl;
   //         }
   //     }
   // }
};
```

lower-upper-bound-flow.hpp

md5: 278a5a

```
template<typename T> struct lrFlow {
   Dinic<T> flow;
   vector<T> in, up;
   int X, Y, n;
   T sum;
   typename Dinic<T>::edge *p, *q;

   lrFlow(int n) : n(n), X(n), Y(n + 1), sum(0), in(n), flow(n
+ 2) {}

   void add_edge(int from, int to, T low, T high) {
      flow.add_edge(from, to, high - low, si(up));
      in[from] -= low, in[to] += low;
      up.eb(high);
   }

   void build() {
      rep(i, n) {
         if(in[i] > 0) {
            flow.add_edge(X, i, in[i]);
            sum += in[i];
         } else if(in[i] < 0) {
            flow.add_edge(i, Y, -in[i]);
         }
      }
   bool can_flow(int s, int t) {
      flow.add_edge(t, s, flow.INF);
      p = &flow.g[t].back();
      q = &flow.g[s].back();
      return can_flow();
   }

   bool can_flow() {
      build();
      auto ret = flow.max_flow(X, Y);
      return ret >= sum;
   }

   T max_flow(int s, int t) {
      if(can_flow(s, t)) {
         return flow.max_flow(s, t);
      } else {
         return -1;
      }
   }

   T min_flow(int s, int t) {
      if(can_flow(s, t)) {
         auto ret = flow.INF - p->cap;
         p->cap = q->cap = 0;
         return ret - flow.max_flow(t, s);
      } else {
         return -1;
      }
   }

   // void output(int M) {
   //     vector<flow_t> ans(M);
   //     for(int i = 0; i < flow.graph.size(); i++) {
   //         for(auto &e : flow.graph[i]) {
   //             if(!e.isrev && ~e.idx) ans[e.idx] = up[e.idx]
- e.cap;
   //         }
   //     }
   //     for(auto &p : ans) cout << p << endl;
   // }
};
```

mcf.hpp

md5: 96eeaa

```
struct MCF {
   struct edge {
      int to;
      ll cap, cost;
      int rev;
      bool isrev;
   };
   vector<vector<edge>> g;
   vl pot, cost;
   vi pv, pe;
   MCF(int n) : g(n) {}
   void add(int u, int v, ll cap, ll cost) {
      g[u].eb(v, cap, cost, si(g[v]), false);
      g[v].eb(u, 0, -cost, si(g[u]) - 1, true);
   }
   ll mcf(int s, int t, ll f) {
      int n = si(g);
      ll ret = 0;
      using P = pair<ll, int>;
      priority_queue<P, vector<P>, greater<P>> pq;
      pot.assign(n, 0), pe.assign(n, -1), pv.assign(n, -1);
      while(f) {
         cost.assign(n, INFL);
         pq.emplace(0, s);
         cost[s] = 0;
         while(!pq.empty()) {
            auto [c, x] = pq.top();
            pq.pop();
            if(cost[x] < c) continue;
            rep(i, si(g[x])) {
               edge& e = g[x][i];
               ll ncost = cost[x] + e.cost + pot[x] -
pot[e.to];
               if(e.cap and chmin(cost[e.to], ncost)) {
                  pv[e.to] = x, pe[e.to] = i;
                  pq.emplace(cost[e.to], e.to);
               }
            }
         }
         if(cost[t] == INFL) return -1;
         rep(i, n) pot[i] += cost[i];
         ll addflow = f;
         for(int v = t; v != s; v = pv[v]) chmin(addflow,
g[pv[v]][pe[v]].cap);
         f -= addflow;
         ret += addflow * pot[t];
         for(int v = t; v != s; v = pv[v]) {
            auto& e = g[pv[v]][pe[v]];
            e.cap -= addflow;
            g[v][e.rev].cap += addflow;
         }
      }
      return ret;
   }
};
```

二部グラフ.md

| | サイズ | 構成| |最大マッチング| $|M|$ | | |最小点被覆| $|M|$ | | L 到達不可能 + R 到達可能| | 最大安定集合| $|V| - |M|$ | | 上の補グラフ| |最小辺被覆| 孤立点がないなら $|V| - |M|$ | 最大マッチング + 含まれない点 greedy|

燃やす埋める.md

変形前の制約	変形後の制約
x が 0 のとき z 失う	(x, T, z)
x が 0 のとき z 得る	無条件で z 得る; (S, x, z)
x が 1 のとき z 失う	(S, x, z)
x が 1 のとき z 得る	無条件で z 得る; (x, T, z)
$x, y, \ldots$ がすべて 0 のとき z 得る	無条件で z 得る; $(S, w, z), (w, x, \infty), (w, y, \infty)$
$x, y, \ldots$ がすべて 1 のとき z 得る	無条件で z 得る; $(w, T, z), (x, w, \infty), (y, w, \infty)$

string

KMP.hpp
md5: 886c63

```
// kmp[i] := max{ l ≤ i | s[:l] == s[(i+1)-l:i+1] }
// abacaba -> 0010123
auto KMP(string s) {
   vector<ll> p(sz(s));
   rep(i, 1, sz(s)) {
      ll g = p[i - 1];
      while(g && s[i] != s[g]) g = p[g - 1];
      p[i] = g + (s[i] == s[g]);
   }
   return p;
}
```

Manacher.hpp
md5: 5882fb

```
// 各位置での回文半径を求める
// aaabaaa -> 1214121
// 偶数長の回文を含めて直径を知るには，N+1 個の $ を挿入して 1 を引く
// $a$a$a$b$a$a$a$ -> 123432181234321
auto manacher(string s) {
   ll n = sz(s), i = 0, j = 0;
   vector<ll> r(n);
   while(i < n) {
      while(i >= j && i + j < n && s[i - j] == s[i + j]) j++;
      r[i] = j;
      ll k = 1;
      while(i >= k && i + k < n && k + r[i - k] < j) {
         r[i + k] = r[i - k];
         k++;
      }
      i += k, j -= k;
   }
   return r;
}
```

RollingHash.hpp
md5: b0e4a8

```
const ll mod = (1LL << 61) - 1;
ll add(ll a, ll b) { return (a += b) >= mod ? a - mod : a; }
ll mul(ll a, ll b) {
   i128 c = (i128)a * b;
   return add(c >> 61, c & mod);
}
ll r = 7954398468495;
struct RH {
   ll n;
   vl hs, pw;
   RH(string s) : n(si(s)), hs(n + 1), pw(n + 1, 1) {
      rep(i, n) {
         pw[i + 1] = mul(pw[i], r);
         hs[i + 1] = add(mul(hs[i], r), s[i]);
      }
   }
   ll get(ll l, ll r) const { return add(hs[r], mod -
mul(hs[l], pw[r - l])); }
   int lcp(int i, int j) {
      int ok = 0, ng = min(n - i, n - j) + 1;
      while(ok < ng - 1) {
         int mid = ok + ng >> 1;
         (get(i, i + mid) == get(j, j + mid) ? ok : ng) = mid;
      }
      return ok;
   }
};
```

SuffixArray.hpp
md5: deae26

```
// returns pair{sa, lcp}
// sa 長さ n : s[sa[0]:] < s[sa[1]:] < … < s[sa[n-1]:]
// lcp 長さ n-1 : lcp[i] = LCP(s[sa[i]:], s[sa[i+1]:])
auto SA(string s) {
   ll n = si(s) + 1, lim = 256;
   // assert(lim > ranges::max(s));
   vector<ll> sa(n), lcp(n), x(all(s) + 1), y(n), ws(max(n,
lim)), rk(n);
   iota(all(sa), 0);
   for(ll j = 0, p = 0; p < n; j = max(1LL, j * 2), lim = p) {
      p = j;
      iota(all(y), n - j);
      rep(i, 0, n) if(sa[i] >= j) y[p++] = sa[i] - j;
      fill(all(ws), 0);
      rep(i, 0, n) ws[x[i]]++;
      rep(i, 1, lim) ws[i] += ws[i - 1];
      for(ll i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
      swap(x, y);
      p = 1;
      x[sa[0]] = 0;
      rep(i, 1, n) {
         ll a = sa[i - 1], b = sa[i];
         x[b] = (y[a] == y[b] && y[a + j] == y[b + j]) ? p - 1
: p++;
      }
   }
   rep(i, 1, n) rk[sa[i]] = i;
   for(ll i = 0, k = 0; i < n - 1; lcp[rk[i++]] = k) {
      if(k) k--;
      while(s[i + k] == s[sa[rk[i] - 1] + k]) k++;
   }
   sa.erase(begin(sa));
   lcp.erase(begin(lcp));
   return pair{sa, lcp};
}
```

Zalgorithm.hpp
md5: d3bdab

```
template<typename T> vi z_algorithm(const vector<T>& s) {
   int n = si(s), l = -1, r = -1;
   vi z(n, n);
   rep(i, 1, n) {
      int& x = z[i] = i < r ? min<ll>(r - i, z[i - l]) : 0;
      while(i + x < n and s[i + x] == s[x]) x++;
      if(i + x > r) l = i, r = i + x;
   }
```

```
    return z;
}
```

enumerate-runs.hpp

md5: aec96b

```
// (length, l, r)
template<typename T> vector<array<int, 3>> enum_runs(const
vector<T>& s) {
    int n = si(s);
    vector<array<int, 3>> res;
    auto dfs = [&](auto&& f, int l, int r) -> void {
        if(r - l <= 1) return;
        int m = l + r >> 1;
        f(f, l, m), f(f, m, r);
        vector<T> sl(s.rbegin() + n - m, s.rbegin() + n - l);
        sl.insert(sl.end(), s.rbegin() + n - r, s.rbegin() + n -
l);
        vector<T> sr(s.begin() + m, s.begin() + r);
        sr.insert(sr.end(), s.begin() + l, s.begin() + r);
        auto zsl = z_algorithm(sl), zsr = z_algorithm(sr);
        rep(t, 1, m - l + 1) {
            int ml = max<ll>(l, m - t - zsl[t]), mr = min(r, m +
zsr[r - l - t]);
            if(mr - ml >= 2 * t and (ml == 0 or s[ml - 1] != s[ml
+ t - 1]) and (mr == n or s[mr] != s[mr - t]))
                res.push_back({ml, mr, t});
        }
        for(int t = 1; t <= r - m; t++) {
            int ml = max(l, m - zsl[r - l - t]), mr = min(r, m + t
+ zsr[t]);
            if(mr - ml >= 2 * t and (ml == 0 or s[ml - 1] != s[ml
+ t - 1]) and (mr == n or s[mr] != s[mr - t]))
                res.push_back({ml, mr, t});
        }
    };
    dfs(dfs, 0, n);
    sort(all(res));
    vector<array<int, 3>> nres;
    int pl = -1, pr = -1;
    for(auto [l, r, t] : res) {
        if(l == pl and r == pr) continue;
        pl = l, pr = r;
        nres.push_back({t, l, r});
    }
    return nres;
}
```

geometry

argument-sort.hpp

md5: 26b1fa

```
bool operator<(point P, point Q) {
    long long C = cross(P, Q);
    if(C == 0 && dot(P, Q) > 0) {
        return false;
    } else if(P.x < 0 && P.y == 0) {
        return true;
    } else if(Q.x < 0 && Q.y == 0) {
        return false;
    } else if(P.y * Q.y <= 0) {
        return P.y < Q.y;
    } else {
        return C > 0;
    }
}
```

circle.hpp

md5: 514ea6

```
struct circle {
    point C;
    double r;
    circle() {}
    circle(point C, double r) : C(C), r(r) {}
};
pair<point, point> line_circle_intersection(line L, circle C) {
    point P = projection(C.C, L);
    double d = point_line_distance(C.C, L);
    double h = sqrt(C.r * C.r - d * d);
    point A = P + vec(L) / abs(vec(L)) * h;
    point B = P - vec(L) / abs(vec(L)) * h;
    return make_pair(A, B);
}
pair<point, point> circle_intersection(circle C1, circle C2) {
    double d = dist(C1.C, C2.C);
    double m = (C1.r * C1.r - C2.r * C2.r + d * d) / (d * 2);
    point M = C1.C + (C2.C - C1.C) / d * m;
    double h = sqrt(C1.r * C1.r - m * m);
    point H = rotate90(C2.C - C1.C) / d * h;
    return make_pair(M - H, M + H);
}
pair<point, point> circle_tangent(point P, circle C) {
    double d = dist(P, C.C);
    double r = sqrt(d * d - C.r * C.r);
    return circle_intersection(C, circle(P, r));
}
vector<line> common_tangent(circle C1, circle C2) {
    if(C1.r < C2.r) { swap(C1, C2); }
    double d = dist(C1.C, C2.C);
    vector<line> L;
    if(C1.r - C2.r <= d + eps) {
        if(C1.r - C2.r <= eps) {
            point D = rotate90(C2.C - C1.C) / d * C1.r;
            L.push_back(line(C1.C + D, C2.C + D));
            L.push_back(line(C1.C - D, C2.C - D));
        } else {
            double m = (C1.r - C2.r) * (C1.r - C2.r) / d;
            point M = C1.C + (C2.C - C1.C) / d * m;
            double h = sqrt((C1.r - C2.r) * (C1.r - C2.r) - m *
m);
            point H1 = M + rotate90(C2.C - C1.C) / d * h;
            point D1 = (H1 - C1.C) / dist(H1, C1.C) * C2.r;
            L.push_back(line(H1 + D1, C2.C + D1));
            point H2 = M - rotate90(C2.C - C1.C) / d * h;
            point D2 = (H2 - C1.C) / dist(H2, C1.C) * C2.r;
            L.push_back(line(H2 + D2, C2.C + D2));
        }
    }
    if(C1.r + C2.r <= d + eps) {
        double m = (C1.r + C2.r) * (C1.r + C2.r) / d;
        point M = C1.C + (C2.C - C1.C) / d * m;
        double h = sqrt((C1.r + C2.r) * (C1.r + C2.r) - m * m);
        point H1 = M + rotate90(C2.C - C1.C) / d * h;
        point D1 = (H1 - C1.C) / dist(H1, C1.C) * C2.r;
        L.push_back(line(H1 - D1, C2.C - D1));
        point H2 = M - rotate90(C2.C - C1.C) / d * h;
        point D2 = (H2 - C1.C) / dist(H2, C1.C) * C2.r;
        L.push_back(line(H2 - D2, C2.C - D2));
    }
    return L;
}
```

convex-hull.hpp

md5: 7b7e26

```
Points convex_hull(Points& p) {
    int n = p.size(), k = 0;
    if(n <= 2) return p;
    sort(begin(p), end(p), [](pt x, pt y) { return (x.x != y.x ?
x.x < y.x : x.y < y.y); });
    Points ch(2 * n);
    for(int i = 0; i < n; ch[k++] = p[i++]) {
        while(k >= 2 && cross(ch[k - 1] - ch[k - 2], p[i] - ch[k
- 1]) <= 0) --k;
    }
    for(int i = n - 2, t = k + 1; i >= 0; ch[k++] = p[i--]) {
        while(k >= t && cross(ch[k - 1] - ch[k - 2], p[i] - ch[k
- 1]) <= 0) --k;
    }
    ch.resize(k - 1);
    return ch;
}
```

funcs.hpp

md5: 19bea4

```
int contains(const Polygon& Q, const Point& p) {
    bool in = false;
    for(int i = 0; i < Q.size(); i++) {
        Point a = Q[i] - p, b = Q[(i + 1) % Q.size()] - p;
        if(a.y > b.y) swap(a, b);
        if(a.y <= 0 && 0 < b.y && cross(a, b) < 0) in = !in;
        if(cross(a, b) == 0 && dot(a, b) <= 0) return _ON;
    }
    return in ? _IN : _OUT;
}

Polygon Minkowski_sum(const Polygon& P, const Polygon& Q) {
   vector<Segment> e1(P.size()), e2(Q.size()), ed(P.size() +
Q.size());
   const auto cmp = [](const Segment& u, const Segment& v) {
return (u.b - u.a).arg_cmp(v.b - v.a); };
   rep(i, P.size()) e1[i] = {P[i], P[(i + 1) % P.size()]};
   rep(i, Q.size()) e2[i] = {Q[i], Q[(i + 1) % Q.size()]};
   rotate(begin(e1), min_element(all(e1), cmp), end(e1));
   rotate(begin(e2), min_element(all(e2), cmp), end(e2));
   merge(all(e1), all(e2), begin(ed), cmp);
   const auto check = [](const Points& res, const Point& u) {
      const auto back1 = res.back(), back2 = *prev(end(res),
2);
      return eq(cross(back1 - back2, u - back2), eps) and
dot(back1 - back2, u - back1) >= -eps;
   };
   auto u = e1[0].a + e2[0].a;
   Points res{u};
   res.reserve(P.size() + Q.size());
   for(const auto& v : ed) {
      u = u + v.b - v.a;
      while(si(res) >= 2 and check(res, u)) res.pop_back();
      res.eb(u);
   }
   if(res.size() and check(res, res[0])) res.pop_back();
   return res;
}

// -1 : on, 0 : out, 1 : in
// O(log(n))
bool is_in(const Polygon& p, const Point& a) {
   if(p.size() == 1) return a == p[0] ? -1 : 0;
   if(p.size() == 2) return intersect(Segment(p[0], p[1]), a);
   if(a == p[0]) return -1;
   if((p[1] - p[0]).toleft(a - p[0]) == -1 || (p.back() -
p[0]).toleft(a - p[0]) == 1) return 0;
   const auto cmp = [&](const Point& u, const Point& v) {
return (u - p[0]).toleft(v - p[0]) == 1; };
   const size_t i = lower_bound(p.begin() + 1, p.end(), a, cmp)
- p.begin();
   if(i == 1) return intersect(Segment(p[0], p[i]), a) ? -1 :
0;
   if(i == p.size() - 1 && intersect(Segment(p[0], p[i]), a))
return -1;
   if(intersect(Segment(p[i - 1], p[i]), a)) return -1;
   return (p[i] - p[i - 1]).toleft(a - p[i - 1]) > 0;
}

using speP = pair<ld, int>;
struct ccut {
   private:
   set<speP> ags;
   vector<int> nexs;
   vector<int> pres;
   vector<Point> ps;

   public:
   void init() {
      const ld sup = -100000;
      ps.push_back({-sup, -sup});
      ps.push_back({sup, -sup});
      ps.push_back({sup, sup});
      ps.push_back({-sup, sup});
      nexs.resize(4);
      pres.resize(4);
      rep(i, 4) {
         int ni = (i + 1) % 4;
         Point dif = ps[ni] - ps[i];
         ld t = arg(dif);
         ags.insert({t, i});
         nexs[i] = ni;
         pres[ni] = i;
      }
   }
   void convex_cut(Point a, Point b) {
      if(ags.empty()) return;
      Point dif = b - a;
      ld t = arg(dif);
      auto itr = ags.lower_bound({t, -1});
      if(itr == ags.end()) itr = ags.begin();
      int cur = (*itr).second;
      if(ccw(a, b, ps[cur]) != -1) return;
      int ricur = nexs[cur];
      while(ricur != cur && ccw(a, b, ps[ricur]) != 1) { ricur
= nexs[ricur]; }
      if(ricur == cur) {
         ags.clear();
         return;
      }
      int lecur = pres[cur];
      while(ccw(a, b, ps[lecur]) != 1) { lecur = pres[lecur]; }
      // new point
      Line l = {a, b};
      Line l1 = {ps[lecur], ps[nexs[lecur]]};
      Line l2 = {ps[pres[ricur]], ps[ricur]};
      Point p1 = is_ll(l1, l);
      Point p2 = is_ll(l2, l);
      int id1 = ps.size();
      int id2 = ps.size() + 1;
      ps.push_back(p1), ps.push_back(p2);
      rep(2) nexs.push_back(-1), pres.push_back(-1);

      // erase(lecur,ricur)
      cur = lecur;
      int tmp = 0;
      while(cur != ricur || !tmp) {
         Point dif = ps[nexs[cur]] - ps[cur];
         ld t = arg(dif);
         ags.erase({t, cur});
         cur = nexs[cur];
         tmp++;
      }

      nexs[lecur] = id1, pres[id1] = lecur, nexs[id1] = id2;
      pres[id2] = id1, nexs[id2] = ricur, pres[ricur] = id2;
      cur = lecur, tmp = 0;
      while(cur != ricur || !tmp) {
         Point dif = ps[nexs[cur]] - ps[cur];
         ld t = arg(dif);
         ags.insert({t, cur});
         cur = nexs[cur];
         tmp++;
      }
   }
   polygon nw_poly() {
      polygon nw;
      for(auto p : ags) nw.push_back(ps[p.second]);
      return nw;
   }
   ld calc_area() {
      polygon nw;
      for(auto p : ags) nw.push_back(ps[p.second]);
      return area(nw);
   }
};
```

line.hpp

md5: 447fab

```
bool point_on_segment(point P, line L) { return dot(P - L.A,
vec(L)) > -eps && dot(P - L.B, vec(L)) < eps; }
point projection(point P, line L) { return L.A + vec(L) /
abs(vec(L)) * dot(P - L.A, vec(L)) / abs(vec(L)); }
point reflection(point P, line L) { return projection(P, L) * 2
- P; }
double point_line_distance(point P, line L) { return
abs(cross(P - L.A, vec(L))) / abs(vec(L)); }
double point_segment_distance(point P, line L) {
   if(dot(P - L.A, vec(L)) < 0) {
      return dist(P, L.A);
   } else if(dot(P - L.B, vec(L)) > 0) {
      return dist(P, L.B);
   } else {
      return point_line_distance(P, L);
   }
}
bool is_parallel(line L1, line L2) { return abs(cross(vec(L1),
vec(L2))) < eps; }
point line_intersection(line L1, line L2) {
   return L1.A + vec(L1) * cross(L2.A - L1.A, vec(L2)) /
cross(vec(L1), vec(L2));
}
bool segment_intersect(line L1, line L2) {
   return cross(L1.A - L2.A, vec(L2)) * cross(L1.B - L2.A,
vec(L2)) < eps
```

```
         && cross(L2.A - L1.A, vec(L1)) * cross(L2.B - L1.A,
vec(L1)) < eps;
}
double segment_distance(line L1, line L2) {
   if(segment_intersect(L1, L2)) {
      return 0;
   } else {
      double ans = INF;
      ans = min(ans, point_segment_distance(L1.A, L2));
      ans = min(ans, point_segment_distance(L1.B, L2));
      ans = min(ans, point_segment_distance(L2.A, L1));
      ans = min(ans, point_segment_distance(L2.B, L1));
      return ans;
   }
}
```

misc

clock.hpp

md5: a1f32c

```
struct Timer {
#define C chrono::high_resolution_clock
   C::time_point c;
   Timer() : c(C::now()) {}
   long long elapsed() {
      auto d = C::now();
      return chrono::duration_cast<chrono::milliseconds>(d -
c).count();
   }
#undef C
};
```

simplex.hpp

md5: 644ba1

```
template<typename F = double, int DEPS = 30, bool Randomize =
true> struct Simplex {
   const F EPS = F(1.0) / (1LL << DEPS);
   int n, m;
   vi shuffle_idx;
   vi idx;
   vector<vector<F>> mat;
   int i_ch, j_ch;

   private:
   void _initialize(const vector<vector<F>>& A, const
vector<F>& b, const vector<F>& c) {
      n = c.size(), m = A.size();

      mat.assign(m + 2, vector<F>(n + 2));
      i_ch = m;
      rep(i, m) {
         rep(j, n) mat[i][j] = -A[i][j];
         mat[i][n] = 1, mat[i][n + 1] = b[i];
         if(mat[i_ch][n + 1] > mat[i][n + 1]) i_ch = i;
      }
      rep(j, n) mat[m][j] = c[j];
      mat[m + 1][n] = -1;

      idx.resize(n + m + 1);
      iota(idx.begin(), idx.end(), 0);
   }

   inline F abs_(F x) noexcept { return x > -x ? x : -x; }
   void _solve() {
      vi jupd;
      for(nb_iter = 0, j_ch = n;; nb_iter++) {
         if(i_ch < m) {
            swap(idx[j_ch], idx[i_ch + n + 1]);
            mat[i_ch][j_ch] = F(1) / mat[i_ch][j_ch];
            jupd.clear();
            rep(j, n + 2) {
               if(j != j_ch) {
                  mat[i_ch][j] *= -mat[i_ch][j_ch];
                  if(abs_(mat[i_ch][j]) > EPS)
jupd.push_back(j);
               }
            }
            rep(i, m + 2) {
               if(abs_(mat[i][j_ch]) < EPS or i == i_ch)
continue;
               fore(j, jupd) mat[i][j] += mat[i][j_ch] *
mat[i_ch][j];
                  mat[i][j_ch] *= mat[i_ch][j_ch];
               }
            }

            j_ch = -1;
            rep(j, n + 1) {
               if(j_ch < 0 or idx[j_ch] > idx[j]) {
                  if(mat[m + 1][j] > EPS or (abs_(mat[m + 1][j]) <
EPS and mat[m][j] > EPS)) j_ch = j;
               }
            }
            if(j_ch < 0) break;

            i_ch = -1;
            rep(i, m) {
               if(mat[i][j_ch] < -EPS) {
                  if(i_ch < 0) {
                     i_ch = i;
                  } else if(mat[i_ch][n + 1] / mat[i_ch][j_ch] -
mat[i][n + 1] / mat[i][j_ch] < -EPS) {
                     i_ch = i;
                  } else if(mat[i_ch][n + 1] / mat[i_ch][j_ch] -
mat[i][n + 1] / mat[i][j_ch] < EPS
                          and idx[i_ch] > idx[i]) {
                     i_ch = i;
                  }
               }
            }
            if(i_ch < 0) {
               is_infty = true;
               break;
            }
         }
         if(mat[m + 1][n + 1] < -EPS) {
            infeasible = true;
            return;
         }
         x.assign(n, 0);
         rep(i, m) {
            if(idx[n + 1 + i] < n) x[idx[n + 1 + i]] = mat[i][n +
1];
         }
         ans = mat[m][n + 1];
      }

      public:
      Simplex(vector<vector<F>> A, vector<F> b, vector<F> c) {
         is_infty = infeasible = false;

         if(Randomize) {
            mt19937
rng(chrono::steady_clock::now().time_since_epoch().count());

            vector<pair<vector<F>, F>> Abs;
            rep(i, si(A)) Abs.emplace_back(A[i], b[i]);
            shuffle(Abs.begin(), Abs.end(), rng);
            A.clear(), b.clear();
            fore(Ab, Abs) A.emplace_back(Ab.first),
b.emplace_back(Ab.second);

            shuffle_idx.resize(c.size());
            iota(all(shuffle_idx), 0);
            shuffle(all(shuffle_idx), rng);
            auto Atmp = A;
            auto ctmp = c;
            rep(i, si(A)) rep(j, si(A[i])) A[i][j] = Atmp[i]
[shuffle_idx[j]];
            rep(j, si(c)) c[j] = ctmp[shuffle_idx[j]];
         }

         _initialize(A, b, c);
         _solve();

         if(Randomize and x.size() == c.size()) {
            auto xtmp = x;
            rep(j, si(c)) x[shuffle_idx[j]] = xtmp[j];
         }
      }
      unsigned nb_iter;
      bool is_infty;
      bool infeasible;
      vector<F> x;
```

```
  static void dual(vector<vector<F>>& A, vector<F>& b,
vector<F>& c) {
      const int n = b.size(), m = c.size();
      vector<vector<F>> At(m, vector<F>(n));
      rep(i, n) rep(j, m) At[j][i] = -A[i][j];
      A = At;
      rep(i, n) b[i] = -b[i];
      rep(j, m) c[j] = -c[j];
      b.swap(c);
  }
};
```

memo

Primes.md

素数の個数

n	10^2	10^3	10^4	10^5	10^6	10^7	10^8	10^9	10^{10}
$\pi(n)$	25	168	1229	9592	78498	664579	5.76e+6	5.08e+7	4.55e+8

高度合成数

$\le n$	10^3	10^4	10^5	10^6	10^7	10^8	10^9
x	840	7560	83160	720720	8648640	73513440	735134400
$d^0(x)$	32	64	128	240	448	768	1344

$\le n$	10^{10}	10^{11}	10^{12}	10^{13}	10^{14}	10^{15}	10^{16}	10^{17}	10^{18}
$d^0(x)$	2304	4032	6720	10752	17280	26880	41472	64512	103680

素数階乗

n	2	3	5	7	11	13	17	19	23	29
$n\#$	2	6	30	210	2310	30030	510510	9.70e+6	2.23e+8	6.47e+9

階乗

4!	5!	6!	7!	8!	9!	10!	11!	12!	13!
24	120	720	5040	40320	362880	3.63e+6	3.99e+7	4.79e+8	6.23e+9

math.md

二項係数

n\k	0	1	2	3	4	5	6	7	8	9	10
0	1										
1	1	1									
2	1	2	1								
3	1	3	3	1							
4	1	4	6	4	1						
5	1	5	10	10	5	1					
6	1	6	15	20	15	6	1				
7	1	7	21	35	35	21	7	1			
8	1	8	28	56	70	56	28	8	1		
9	1	9	36	84	126	126	84	36	9	1	
10	1	10	45	120	210	252	210	120	45	10	1
11	1	11	55	165	330	462	462	330	165	55	11
12	1	12	66	220	495	792	924	792	495	220	66
13	1	13	78	286	715	1287	1a716	1716	1287	715	286
14	1	14	91	364	1001	2002	3003	3432	3003	2002	1001
15	1	15	105	455	1365	3003	5005	6435	6435	5005	3003
16	1	16	120	560	1820	4368	8008	11440	12870	11440	8008
17	1	17	136	680	2380	6188	12376	19448	24310	24310	19448
18	1	18	153	816	3060	8568	18564	31824	43758	48620	43758
19	1	19	171	969	3876	11628	27132	50388	75582	92378	92378

$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$ $\binom{L}{k} + \dots + \binom{R-1}{k} = \binom{R}{k+1} - \binom{L}{k+1}$

第一種スターリング数

$c(n,k)$: $1,2,\dots,n$ の順列で巡回置換 k 個に分割できるものの個数

$n \setminus k$	0	1	2	3	4	5	6	7
0	1							
1	0	1						
2	0	1	1					
3	0	2	3	1				
4	0	6	11	6	1			
5	0	24	50	35	10	1		
6	0	120	274	225	85	15	1	
7	0	720	1764	1624	735	175	21	1

$c(n,k) = c(n-1,k-1) + (n-1)c(n-1,k)$
$x(x+1)\dots(x+n-1) = \sum_{k=0}^{n} c(n,k)x^k$ $\sum_{k=0}^{n} c(n,k) = n!$
$\sum_{k=0}^{n} 2^k c(n,k) = (n+1)!$ $\sum_{k=0}^{n}(-1)^k c(n,k) = 0$

$\sum_{k=0}^{n} c(n,k)x^k = x(x+1)\dots(x+n-1)$ を用いて分割統治し，片方の計算を polynomial taylor shift で再利用すると，$c(N,k)$ の k に関する列挙が $O(N\log N)$ 時間でできる．

第二種スターリング数

$S(n,k)$: $1,2,\dots,n$ を k 個の区別しない集合に分割する方法の数

$n \setminus k$	0	1	2	3	4	5	6	7
0	1							
1	0	1						
2	0	1	1					
3	0	1	3	1				
4	0	1	7	6	1			
5	0	1	15	25	10	1		
6	0	1	31	90	65	15	1	
7	0	1	63	301	350	140	21	1

$S(n,k) = S(n-1,k-1) + kS(n-1,k)$
$x^n = \sum_{k=0}^{n} S(n,k)x(x-1)\dots(x-k+1)$
$S(n,k) = \frac{1}{k!}\sum_{m=1}^{k}(-1)^{k-m}\binom{k}{m}m^n$

最後の式と畳み込みを使うと $S(N,k)$ の k に関する列挙が $O(N\log N)$ 時間でできる．

ベル数

B_n: $1,2,\dots,n$ をいくつかの集合に分割する方法の数

n	0	1	2	3	4	5	6	7	8	9	10
B_n	1	1	2	5	15	52	203	877	4140	21147	115975

$B_{n+1} = \sum_{k=0}^{n}\binom{n}{k}B_k$ $B_n = \frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!}$

指数型母関数 $\exp(\exp x - 1) = \sum_{n=0}^{\infty} B_n \frac{x^n}{n!}$ を使うと $B_0, B_1, \dots, B_n$ の計算が $O(N\log N)$ でできる．

カタラン数

C_n: n 個の (と) を括弧列になるように並べる方法の数

n	0	1	2	3	4	5	6	7	8	9	10
C_n	1	1	2	5	14	42	132	429	1430	4862	16796

モンモール数

a_n: $1, 2, \ldots, n$ の順列 P で $P_i \neq i$ となるものの個数

n	0	1	2	3	4	5	6	7	8	9	10
a_n		0	1	2	9	44	265	1854	14833	133496	1334961

$a_n = (n-1)(a_{n-1} + a_{n-2})$ $a_n = na_{n-1} + (-1)^n$

分割数

P_n: n を正の整数の和として表す方法の数

n	0	1	2	3	4	5	6	7	8	9	10
P_n	1	1	2	3	5	7	11	15	22	30	42

母関数は $\prod_{n=1}^{\infty} \frac{1}{1-x^n} = \sum_{n=0}^{\infty} P_n x^n$ である．五角数定理より $\prod_{n=1}^{\infty}(1-x^n) = \sum_{n=-\infty}^{\infty}(-1)^n x^{n(3n+1)/2}$ なので，$P_0, P_1, \ldots, P_N$ を $O(N \log N)$ 時間で列挙できる．

母関数

$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \cdots = \sum_{n=0}^{\infty} x^n$

$\frac{1}{(1-x)^2} = 1 + 2x + 3x^2 + 4x^3 + 5x^4 + \cdots = \sum_{n=0}^{\infty}(n+1)x^n$

$\frac{1}{(1-x)^3} = 1 + 3x + 6x^2 + 10x^3 + 15x^4 + \cdots = \sum_{n=0}^{\infty} \frac{1}{2}(n+1)(n+2)x^n$

$\frac{1}{(1-x)^d} = \sum_{n=0}^{\infty} \binom{n+d-1}{n} x^n$

$\sqrt{1-x} = 1 - \frac{1}{2}x - \frac{1}{8}x^2 - \frac{1}{16}x^3 - \frac{5}{128}x^4 - \cdots = 1 - \sum_{n=1}^{\infty} \frac{(2n-2)!}{2^{2n-1}n!(n-1)!} x^n$

$\frac{1}{\sqrt{1-x}} = 1 + \frac{1}{2}x + \frac{3}{8}x^2 + \frac{5}{16}x^3 + \frac{35}{128}x^4 - \cdots = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n n!} x^n$

$\frac{1-\sqrt{1-4x}}{2x} = 1 + x + 2x^2 + 5x^3 + 14x^4 + \cdots = \sum_{n=0}^{\infty} C_n x^n = \sum_{n=0}^{\infty} \frac{(2n)!}{(n+1)!n!} x^n$ (カタラン数)

$\frac{1}{\sqrt{1-4x}} = \sum_{n=0}^{\infty} \binom{2n}{n} x^n$

$\frac{1}{1-x-x^2} = 1 + x + 2x^2 + 3x^3 + 5x^4 + \cdots = \sum_{n=0}^{\infty} F_n x^n$ (フィボナッチ数)

$\log(1-x) = -x - \frac{1}{2}x^2 - \frac{1}{3}x^3 - \frac{1}{4}x^4 - \cdots = \sum_{n=1}^{\infty} \frac{1}{n} x^n$

$\exp(\exp x - 1) = \sum_{n=0}^{\infty} B_n \frac{x^n}{n}$ (ベル数)

$\frac{1}{k}(\exp x - 1)^k = \sum_{n=0}^{\infty} S(n,k) \frac{x^n}{n!}$ (第二種スターリング数)

$\frac{\exp(-x)}{1-x} = 1 + \frac{1}{2}x^2 + \frac{1}{3}x^3 + \frac{3}{8}x^4 + \cdots = \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}$ (モンモール数)

$C(x)^k = \left(\frac{1-\sqrt{1-4x}}{2x}\right)^k = \sum_{n=0}^{\infty} \frac{k}{n+k} \binom{2n+k-1}{n}$ (カタラン数の母関数の累乗)