ICPC Notebook

template

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
```

settings.sh

```
# CLion の設定
Settings → Build → CMake → Reload CMake Project
add_compile_options(-D_GLIBCXX_DEBUG)
# Caps Lock を Ctrl に変更
setxkbmap -option ctrl:nocaps
```

template.hpp

md5: b929fd

```cpp
#pragma GCC target("avx2")
#pragma GCC optimize("O3")
#pragma GCC optimize("unroll-loops")

#include <bits/stdc++.h>
using namespace std;
using ll = long long;
const ll INF = LLONG_MAX / 4;
#define all(a) begin(a), end(a)
bool chmin(auto& a, auto b) { return a > b ? a = b, 1 : 0; }
bool chmax(auto& a, auto b) { return a < b ? a = b, 1 : 0; }
#define dout(a) cout << fixed << setprecision(10) << a << endl;

int main() {
   cin.tie(0)->sync_with_stdio(0);
   // your code here...
}
```

data-structure

2d_segtree.cpp

md5: 5be105

```cpp
template<typename T, typename F> struct SegmentTree2D {
   private:
   int id(int h, int w) const { return h * 2 * W + w; }

   public:
   int H, W;
   vector<T> seg;
   const F f;
   const T I;

   SegmentTree2D(int h, int w, F _f, const T& i) : f(_f), I(i) { init(h, w); }

   void init(int h, int w) {
      H = W = 1;
      while(H < h) H <<= 1;
      while(W < w) W <<= 1;
      seg.assign(4 * H * W, I);
   }

   void set(int h, int w, const T& x) { seg[id(h + H, w + W)] = x; }

   void build() {
      for(int w = W; w < 2 * W; w++) {
         for(int h = H - 1; h; h--) { seg[id(h, w)] = f(seg[id(2 * h + 0, w)], seg[id(2 * h + 1, w)]); }
      }
      for(int h = 0; h < 2 * H; h++) {
         for(int w = W - 1; w; w--) { seg[id(h, w)] = f(seg[id(h, 2 * w + 0)], seg[id(h, 2 * w + 1)]); }
      }
   }

   T get(int h, int w) const { return seg[id(h + H, w + W)]; }
   T operator()(int h, int w) const { return seg[id(h + H, w + W)]; }

   void update(int h, int w, const T& x) {
```

```
      h += H, w += W;
      seg[id(h, w)] = x;
      for(int i = h >> 1; i; i >>= 1) { seg[id(i, w)] = f(seg[id(2
* i + 0, w)], seg[id(2 * i + 1, w)]); }
      for(; h; h >>= 1) {
         for(int j = w >> 1; j; j >>= 1) { seg[id(h, j)] =
f(seg[id(h, 2 * j + 0)], seg[id(h, 2 * j + 1)]); }
      }
   }

   T _inner_query(int h, int w1, int w2) {
      T res = I;
      for(; w1 < w2; w1 >>= 1, w2 >>= 1) {
         if(w1 & 1) res = f(res, seg[id(h, w1)]), w1++;
         if(w2 & 1) --w2, res = f(res, seg[id(h, w2)]);
      }
      return res;
   }

   T query(int h1, int w1, int h2, int w2) {
      if(h1 >= h2 || w1 >= w2) return I;
      T res = I;
      h1 += H, h2 += H, w1 += W, w2 += W;
      for(; h1 < h2; h1 >>= 1, h2 >>= 1) {
         if(h1 & 1) res = f(res, _inner_query(h1, w1, w2)), h1++;
         if(h2 & 1) --h2, res = f(res, _inner_query(h2, w1, w2));
      }
      return res;
   }
};
```

beats.md

lazy_segtreeの一部を変更して作る

```
void all_apply(int k, F f) {
    d[k] = mappin(f, d[k]);
-   if(k < size) lz[k] = composition(f, lz[k]);
+   if(k < size)){
+       lz[k] = composition(f, lz[k]);
+       if(d[k].fail)push(k), update(k);
+   }
}
```

- Sはalcoder::lazy_segtree空参照可能なメンバ変数failを持つ
- mapping関数によるSの元xへの作用の結果を得る計算が(xの持つ情報の不足が原因で)失敗した時のみ，mapping関数が返すSのインスタンスのfailの値はtrueとなる
- mapping関数による作用以外の部分で計算は失敗しない
- 要素数1の区間を管理するSの元に対するmappingは失敗してはならない

beats_example.cpp md5: eb71ce

```
// 区間chmax, chmin, add 区間sum取得
namespace RangeChMinMaxAddSum {
template<typename Num> inline Num second_lowest(Num a, Num a2, Num
c, Num c2) noexcept {
   // a < a2, c < c2 のとき全引数を昇順ソートして二番目の値
   return a == c ? std::min(a2, c2) : a2 <= c ? a2 : c2 <= a ? c2 :
std::max(a, c);
}
template<typename Num> inline Num second_highest(Num a, Num a2, Num
b, Num b2) noexcept {
   // a > a2, b > b2 のとき全引数を降順ソートして二番目の値
   return a == b ? std::max(a2, b2) : a2 >= b ? a2 : b2 >= a ? b2 :
std::min(a, b);
}

constexpr ll BINF = 1LL << 61;

struct S {
   ll lo, hi, lo2, hi2, sum;  // 区間最小・最大値，区間最小・最大から二番
目の値，区間総和
   unsigned sz, nlo, nhi;     // 区間要素数，区間最小・最大値をとる要素の
個数
   bool fail;
   S() : lo(BINF), hi(-BINF), lo2(BINF), hi2(-BINF), sum(0), sz(0),
nlo(0), nhi(0), fail(0) {}
   S(ll x, unsigned sz_ = 1)
       : lo(x), hi(x), lo2(BINF), hi2(-BINF), sum(x * sz_),
sz(sz_), nlo(sz_), nhi(sz_), fail(0) {}
};

S e() { return S(); }

S op(S l, S r) {
   S ret;
   ret.lo = min(l.lo, r.lo), ret.hi = max(l.hi, r.hi);
   ret.lo2 = second_lowest(l.lo, l.lo2, r.lo, r.lo2);
   ret.hi2 = second_highest(l.hi, l.hi2, r.hi, r.hi2);
   ret.sum = l.sum + r.sum, ret.sz = l.sz + r.sz;
   ret.nlo = l.nlo * (l.lo <= r.lo) + r.nlo * (r.lo <= l.lo);
   ret.nhi = l.nhi * (l.hi >= r.hi) + r.nhi * (r.hi >= l.hi);
   return ret;
}

struct F {
   ll lb, ub, bias;
   F(ll chmax_ = -BINF, ll chmin_ = BINF, ll add = 0) : lb(chmax_),
ub(chmin_), bias(add) {}
   static F chmin(ll x) noexcept { return F(-BINF, x, ll(0)); }
   static F chmax(ll x) noexcept { return F(x, BINF, ll(0)); }
   static F add(ll x) noexcept { return F(-BINF, BINF, x); };
};

F composition(F fnew, F fold) {
   F ret;
   ret.lb = max(min(fold.lb + fold.bias, fnew.ub), fnew.lb) -
fold.bias;
   ret.ub = min(max(fold.ub + fold.bias, fnew.lb), fnew.ub) -
fold.bias;
   ret.bias = fold.bias + fnew.bias;
   return ret;
}

F id() { return F(); }

S mapping(F f, S x) {
   if(x.sz == 0) return e();
   else if(x.lo == x.hi or f.lb == f.ub or f.lb >= x.hi or f.ub <=
x.lo) {
      return S(min(max(x.lo, f.lb), f.ub) + f.bias, x.sz);
   } else if(x.lo2 == x.hi) {
      x.lo = x.hi2 = max(x.lo, f.lb) + f.bias;
      x.hi = x.lo2 = min(x.hi, f.ub) + f.bias;
      x.sum = x.lo * x.nlo + x.hi * x.nhi;
      return x;
   } else if(f.lb < x.lo2 and f.ub > x.hi2) {
      ll nxt_lo = max(x.lo, f.lb), nxt_hi = min(x.hi, f.ub);
      x.sum += (nxt_lo - x.lo) * x.nlo - (x.hi - nxt_hi) * x.nhi +
f.bias * x.sz;
      x.lo = nxt_lo + f.bias, x.hi = nxt_hi + f.bias, x.lo2 +=
f.bias, x.hi2 += f.bias;
      return x;
   }
   x.fail = 1;
   return x;
}
using segtree = lazy_segtree<S, op, e, F, mapping, composition,
id>;
}  // namespace RangeChMinMaxAddSum
```

binary_trie.hpp md5: af8046

```
// base: b75bb1
template<typename T, int MAX_LOG = 32> struct BinaryTrie {
   struct Node {
      array<int, 2> next;
      int common;
      T lazy;
      Node() : next{-1, -1}, common(), lazy() {}
   };
   vector<Node> nodes;
   BinaryTrie() { nodes.push_back(Node()); }
   void apply_xor(T val) { nodes[0].lazy ^= val; }
   void push(int cur, int b) {
      if((nodes[cur].lazy >> b) & 1) swap(nodes[cur].next[0],
nodes[cur].next[1]);
      for(int i = 0; i < 2; i++) {
         if(nodes[cur].next[i] == -1)
nodes[nodes[cur].next[i]].lazy ^= nodes[cur].lazy;
      }
      nodes[cur].lazy = 0;
   }
   void add(T val, int cur = 0, int b = MAX_LOG - 1) {
      nodes[cur].common++;
      if(b == -1) return;
      push(cur, b);
      int nxt = (val >> (T)b) & (T)1;
      if(nodes[cur].next[nxt] == -1) {
         nodes[cur].next[nxt] = size(nodes);
         nodes.push_back(Node());
      }
```

```
      add(val, nodes[cur].next[nxt], b - 1);
   }

   void erase(T val, int cur = 0, int b = MAX_LOG - 1) {
      nodes[cur].common--;
      if(b == -1) return;
      push(cur, b);
      erase(val, nodes[cur].next[(val >> b) & 1], b - 1);
   }

   T min_element(T val = 0, int cur = 0, int b = MAX_LOG - 1) {
      if(b == -1) return 0;
      push(cur, b);
      T nxt = (val >> b) & 1;
      int ind = nodes[cur].next[nxt];
      if(ind == -1 || nodes[ind].common == 0) nxt ^= 1;
      return min_element(val, nodes[cur].next[nxt], b - 1) | (nxt
<< b);
   }  // ddf699

   T max_element(T val = 0, int cur = 0, int b = MAX_LOG - 1) {
return min_element(~val); }  // 5e1a86

   int lower_bound_rank(T val, int cur = 0, int b = MAX_LOG - 1) {
      if(cur == -1 || b == -1) return 0;
      push(cur, b);
      T nxt = (val >> b) & 1;
      int ret = lower_bound_rank(val, nodes[cur].next[nxt], b - 1);
      if(nxt == 1 && nodes[cur].next[0] != -1) { ret +=
nodes[nodes[cur].next[0]].common; }
      return ret;
   }  // 05b14c

   int upper_bound_rank(T val) { return lower_bound_rank(val + 1);
}  // 70e301

   T kth_smallest(int k, int cur = 0, int b = MAX_LOG - 1) {
      if(b == -1) return 0;
      int lower_ind = nodes[cur].next[0];
      int lower_cnt = 0;
      if(lower_ind != -1) lower_cnt = nodes[lower_ind].common;
      if(k < lower_cnt) return kth_smallest(k, nodes[cur].next[0],
b - 1);
      return kth_smallest(k - lower_cnt, nodes[cur].next[1], b - 1)
| (T(1) << b);
   }  // b85845

   T kth_largest(int k) { return kth_smallest(nodes[0].common - k);
}  // 1df41b

   int count(T val) {
      int cur = 0;
      for(int b = MAX_LOG - 1; b >= 0; b--) {
         push(cur, b);
         cur = nodes[cur].next[(val >> b) & 1];
         if(cur == -1) return 0;
      }
      return nodes[cur].common;
   }  // 2a3342

   int count() { return nodes[0].common; }  // 210f0e
};
```

cht_slope_monotone.hpp md5: 720828

```
template<typename T> class CHT {
   using P = pair<T, T>;

   private:
   int head;
   vector<P> lines;
   // 最小値(最大値)を求めるxが単調であるか
   bool isMonotonicX;
   function<bool(T l, T r)> comp;

   public:
   // クエリが単調であった場合はflag = trueとする
   CHT(
       bool flagX = false, function<bool(T l, T r)> compFunc = [](T
l, T r) { return l >= r; })
       : isMonotonicX(flagX), comp(compFunc), head(0){};

   bool check(P l1, P l2, P l3) {
      if(l1 < l3) swap(l1, l3);
      return (l3.second - l2.second) * (l2.first - l1.first) >=
(l2.second - l1.second) * (l3.first - l2.first);
   }
   void add(T a, T b) {
      P line(a, b);
      while(lines.size() >= 2 + head && check(*(lines.end() - 2),
lines.back(), line)) lines.pop_back();
      lines.emplace_back(line);
   }

   T f(int i, T x) { return lines[i].first * x + lines[i].second; }

   T f(P line, T x) { return line.first * x + line.second; }

   T get(T x) {
      if(isMonotonicX) {
         while(lines.size() >= 2 + head && comp(f(head, x), f(head
+ 1, x))) ++head;
         return f(head, x);
      } else {
         int low = -1, high = lines.size() - 1;
         while(high - low > 1) {
            int mid = (high + low) / 2;
            (comp(f(mid, x), f(mid + 1, x)) ? low : high) = mid;
         }
         return f(high, x);
      }
   }
};
```

disjoint_sparse_table.hpp md5: 3df31b

```
template<typename SG> struct disjoint_sparse_table {
   using S = typename SG::S;
   vector<vector<S>> st;
   vector<int> lg;

   disjoint_sparse_table(const vector<S>& v) {
      int b = 0;
      while((1 << b) <= size(v)) b++;
      st.assign(b, vector<S>(size(v)));
      for(int i = 0; i < size(v); i++) st[0][i] = v[i];
      for(int i = 1; i < b; i++) {
         int shift = 1 << i;
         for(int j = 0; j < size(v); j += shift << 1) {
            int t = min(j + shift, (int)size(v));
            st[i][t - 1] = v[t - 1];
            for(int k = t - 2; k >= j; k--) st[i][k] = SG::op(v[k],
st[i][k + 1]);
            if(size(v) <= t) break;
            st[i][t] = v[t];
            for(int k = t + 1; k < min((int)size(v), t + shift);
k++) st[i][k] = SG::op(st[i][k - 1], v[k]);
         }
      }
      lg.resize(1 << b);
      for(int i = 2; i < size(lg); i++) lg[i] = lg[i >> 1] + 1;
   }
   S prod(int l, int r) {
      if(l >= --r) return st[0][l];
      int b = lg[l ^ r];
      return SG::op(st[b][l], st[b][r]);
   }
};
```

lazy_segtree.hpp md5: c86cef

```
// base: 918715
unsigned int bit_ceil(unsigned int n) {
   unsigned int x = 1;
   while(x < (unsigned int)(n)) x *= 2;
   return x;
}
int countr_zero(unsigned int n) { return __builtin_ctz(n); }
constexpr int countr_zero_constexpr(unsigned int n) {
   int x = 0;
   while(!(n & (1 << x))) x++;
   return x;
}
template<class S, S (*op)(S, S), S (*e)(), class F, S (*mapping)(F,
S), F (*composition)(F, F), F (*id)()>
struct lazy_segtree {
   public:
   lazy_segtree() : lazy_segtree(0) {}
   explicit lazy_segtree(int n) : lazy_segtree(vector<S>(n, e()))
{}
   explicit lazy_segtree(const vector<S>& v) : _n(int(v.size())) {
      size = (int)bit_ceil((unsigned int)(_n));
      log = countr_zero((unsigned int)size);
      d = vector<S>(2 * size, e());
      lz = vector<F>(size, id());
      for(int i = 0; i < _n; i++) d[size + i] = v[i];
```

```
    for(int i = size - 1; i >= 1; i--) { update(i); }
  }
  void set(int p, S x) {
    // assert(0 <= p && p < _n);
    p += size;
    for(int i = log; i >= 1; i--) push(p >> i);
    d[p] = x;
    for(int i = 1; i <= log; i++) update(p >> i);
  }

  S get(int p) {
    // assert(0 <= p && p < _n);
    p += size;
    for(int i = log; i >= 1; i--) push(p >> i);
    return d[p];
  }

  S prod(int l, int r) {
    // assert(0 <= l && l <= r && r <= _n);
    if(l == r) return e();

    l += size;
    r += size;

    for(int i = log; i >= 1; i--) {
      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);
  }

  void apply(int l, int r, F f) {
    assert(0 <= l && l <= r && r <= _n);
    if(l == r) return;

    l += size;
    r += size;

    for(int i = log; i >= 1; i--) {
      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) all_apply(l++, f);
        if(r & 1) all_apply(--r, f);
        l >>= 1;
        r >>= 1;
      }
      l = l2;
      r = r2;
    }

    for(int i = 1; i <= log; i++) {
      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(0 <= l && l <= _n);
    // assert(g(e()));
    if(l == _n) return _n;
    l += size;
    for(int i = log; i >= 1; i--) push(l >> i);
    S sm = e();
    do {
      while(l % 2 == 0) l >>= 1;
      if(!g(op(sm, d[l]))) {
        while(l < size) {
          push(l);
          l = (2 * l);
          if(g(op(sm, d[l]))) {
            sm = op(sm, d[l]);
            l++;
          }
        }
        return l - size;
      }
      sm = op(sm, d[l]);
      l++;
    } while((l & -l) != l);
    return _n;
  }  // d93691

  template<class G> int min_left(int r, G g) {
    // assert(0 <= r && r <= _n);
    // assert(g(e()));
    if(r == 0) return 0;
    r += size;
    for(int i = log; i >= 1; i--) push((r - 1) >> i);
    S sm = e();
    do {
      r--;
      while(r > 1 && (r % 2)) r >>= 1;
      if(!g(op(d[r], sm))) {
        while(r < size) {
          push(r);
          r = (2 * r + 1);
          if(g(op(d[r], sm))) {
            sm = op(d[r], sm);
            r--;
          }
        }
        return r + 1 - size;
      }
      sm = op(d[r], sm);
    } while((r & -r) != r);
    return 0;
  }  // c9a7eb

  private:
  int _n, size, 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] = mapping(f, d[k]);
    if(k < size) lz[k] = composition(f, lz[k]);
  }
  void push(int k) {
    all_apply(2 * k, lz[k]);
    all_apply(2 * k + 1, lz[k]);
    lz[k] = id();
  }
};
```

li_chao.hpp

md5: 08f964

```
template<typename T> class LiChaoTree {
  int n, sz, height;
  vector<T> xs, as, bs;

  void update(T a, T b, int k, int h) {
    int l = (k << h) & (sz - 1);
    int r = l + (1 << h);

    while(1) {
      int m = (l + r) >> 1;
      T xl = xs[l], xm = xs[m];
      bool l_update = a * xl + b < as[k] * xl + bs[k];
      bool m_update = a * xm + b < as[k] * xm + bs[k];

      if(m_update) {
        swap(as[k], a);
        swap(bs[k], b);
      }
      if(h == 0) break;
      if(l_update != m_update) {
        k = k * 2;
        r = m;
      } else {
        k = k * 2 + 1;
        l = m;
      }
      h--;
    }
  }

  public:
  // 最小値クエリのx座標や線分追加クエリの両端のx座標を先読みしてソートした配列
  LiChaoTree(const vector<T>& xs) : n(xs.size()), xs(xs) {
    sz = 1, height = 0;
```

```
      while(sz < (int)xs.size()) {
         sz <<= 1;
         height++;
      }
      this->xs.resize(sz, xs.back());
      as.assign(sz * 2, 0);
      bs.assign(sz * 2, std::numeric_limits<T>::max());
   }

   void add_line(T a, T b) { update(a, b, 1, height); }

   void add_segment(T a, T b, int l, int r) {
      if(l == r) return;
      l += sz, r += sz;
      int h = 0;
      while(l < r) {
         if(l & 1) update(a, b, l++, h);
         if(r & 1) update(a, b, --r, h);
         l >>= 1, r >>= 1, h++;
      }
   }

   // x = xs[i] における最小値を求める
   T get(int i) const {
      T x = xs[i];
      int k = i + sz;
      T res = as[k] * x + bs[k];
      while(k > 1) {
         k >>= 1;
         T tmp = as[k] * x + bs[k];
         if(tmp < res) res = tmp;
      }
      return res;
   }
};
```

persistent-array.cpp md5: bb60a8

```
template<typename T, int LOG> struct PersistentArray {
   struct Node {
      T data;
      Node* child[1 << LOG] = {};
      Node() {}
      Node(const T& data) : data(data) {}
   };

   Node* root;
   PersistentArray() : root(nullptr) {}

   T get(Node* t, int k) {
      if(k == 0) return t->data;
      return get(t->child[k & ((1 << LOG) - 1)], k >> LOG);
   }

   T get(const int& k) { return get(root, k); }

   pair<Node*, T*> mutable_get(Node* t, int k) {
      t = t ? new Node(*t) : new Node();
      if(k == 0) return {t, &t->data};
      auto p = mutable_get(t->child[k & ((1 << LOG) - 1)], k >>
LOG);
      t->child[k & ((1 << LOG) - 1)] = p.first;
      return {t, p.second};
   }

   T* mutable_get(const int& k) {
      auto ret = mutable_get(root, k);
      root = ret.first;
      return ret.second;
   }

   Node* build(Node* t, const T& data, int k) {
      if(!t) t = new Node();
      if(k == 0) {
         t->data = data;
         return t;
      }
      auto p = build(t->child[k & ((1 << LOG) - 1)], data, k >>
LOG);
      t->child[k & ((1 << LOG) - 1)] = p;
      return t;
   }

   void build(const vector<T>& v) {
      root = nullptr;
      for(int i = 0; i < v.size(); i++) { root = build(root, v[i],
i); }
   }
};
```

persistent-segtree.cpp md5: b2594f

```
template<typename Monoid> struct PersistentSegmentTree {
   using F = function<Monoid(Monoid, Monoid)>;

   struct Node {
      Monoid data;
      Node *l, *r;

      Node(const Monoid& data) : data(data), l(nullptr), r(nullptr)
{}
   };

   int sz;
   const F f;
   const Monoid M1;

   PersistentSegmentTree(const F f, const Monoid& M1) : f(f),
M1(M1) {}

   Node* build(vector<Monoid>& v) {
      sz = (int)v.size();
      return build(0, (int)v.size(), v);
   }

   Node* merge(Node* l, Node* r) {
      auto t = new Node(f(l->data, r->data));
      t->l = l;
      t->r = r;
      return t;
   }

   Node* build(int l, int r, vector<Monoid>& v) {
      if(l + 1 >= r) return new Node(v[l]);
      return merge(build(l, (l + r) >> 1, v), build((l + r) >> 1,
r, v));
   }

   Node* update(int a, const Monoid& x, Node* k, int l, int r) {
      if(r <= a || a + 1 <= l) return k;
      else if(a <= l && r <= a + 1) return new Node(x);
      else return merge(update(a, x, k->l, l, (l + r) >> 1),
update(a, x, k->r, (l + r) >> 1, r));
   }

   Node* update(Node* t, int k, const Monoid& x) { return update(k,
x, t, 0, sz); }

   Monoid query(int a, int b, Node* k, int l, int r) {
      if(r <= a || b <= l) return M1;
      else if(a <= l && r <= b) return k->data;
      else return f(query(a, b, k->l, l, (l + r) >> 1), query(a, b,
k->r, (l + r) >> 1, r));
   }

   Monoid query(Node* t, int a, int b) { return query(a, b, t, 0,
sz); }
};
```

persistent-uf.cpp md5: 8d972c

```
struct PersistentUnionFind {
   PersistentArray<int, 3> data;

   PersistentUnionFind() {}

   PersistentUnionFind(int sz) { data.build(vector<int>(sz, -1)); }

   int find(int k) {
      int p = data.get(k);
      return p >= 0 ? find(p) : k;
   }

   int size(int k) { return (-data.get(find(k))); }

   bool unite(int x, int y) {
      x = find(x);
      y = find(y);
      if(x == y) return false;
      auto u = data.get(x);
      auto v = data.get(y);

      if(u < v) {
         auto a = data.mutable_get(x);
         *a += v;
         auto b = data.mutable_get(y);
```

```
        *b = x;
      } else {
        auto a = data.mutable_get(y);
        *a += u;
        auto b = data.mutable_get(x);
        *b = y;
      }
      return true;
   }
};
```

potential_dsu.hpp md5: b2e5eb

```
// base: 650ffa
template<typename Abel> struct potential_dsu {
   using T = typename Abel::T;
   int tCount;
   vector<int> p, rank;
   vector<T> potential;
   int N;
   potential_dsu(int size) {
      N = size;
      p.resize(N, -1);
      rank.resize(N, 0);
      potential.resize(N, Abel::e());
      tCount = N;
   }

   bool same(int x, int y) { return leader(x) == leader(y); }

   // w[y] - w[x] = w
   void merge(int x, int y, T w) {
      w = Abel::op(w, get_potential(x));
      w = Abel::op(w, Abel::inv(get_potential(y)));
      link(leader(x), leader(y), w);
   }

   int leader(int x) {
      if(p[x] < 0) return x;
      int l = leader(p[x]);
      potential[x] = Abel::op(potential[x], potential[p[x]]);
      return p[x] = l;
   }

   T get_potential(int x) {
      leader(x);
      return potential[x];
   }

   // w[y] - w[x]
   T diff(int x, int y) { return Abel::op(get_potential(y),
Abel::inv(get_potential(x))); }

   int count() { return tCount; }  // 154012

   int size(int a) {
      // assert(0 <= a && a < _n);
      return -p[leader(a)];
   }  // 818fe7

   vector<vector<int>> groups() {
      vector<int> leader_buf(N), group_size(N);
      for(int i = 0; i < N; i++) {
         leader_buf[i] = leader(i);
         group_size[leader_buf[i]]++;
      }
      vector<vector<int>> result(N);
      for(int i = 0; i < N; i++) result[i].reserve(group_size[i]);
      for(int i = 0; i < N; i++)
result[leader_buf[i]].push_back(i);
      result.erase(remove_if(result.begin(), result.end(), [&]
(const vector<int>& v) { return v.empty(); }),
                   result.end());
      return result;
   }  // 92d7ce

   private:
   void link(int x, int y, T w) {
      if(x == y) return;
      tCount--;
      if(rank[x] < rank[y]) {
         swap(x, y);
         w = Abel::inv(w);
      }
      p[x] += p[y];
      p[y] = x;
      if(rank[x] == rank[y]) rank[x]++;
      tCount--;
      potential[y] = w;
   }
};

/*
struct Abel {
   using T = int;
   static T e() { return 0; }
   static T op(T a, T b) { return a + b; }
   static T inv(T a) { return -a; }
};

potential_dsu<Abel> dsu(N);
*/
```

range_set.hpp md5: 1bc645

```
template<bool margeAdjacent = true> struct range_set : public
map<ll, ll> {
   auto get(ll p) const {
      auto it = upper_bound(p);
      if(it == begin() || (--it)->second < p) return end();
      return it;
   }

   void insert(ll l, ll r) {
      auto itl = upper_bound(l), itr = upper_bound(r +
margeAdjacent);
      if(itl != begin()) {
         if((--itl)->second < l - margeAdjacent) ++itl;
      }
      if(itl != itr) {
         l = min(l, itl->first);
         r = max(r, prev(itr)->second);
         erase(itl, itr);
      }
      (*this)[l] = r;
   }

   void remove(ll l, ll r) {
      auto itl = upper_bound(l), itr = upper_bound(r);
      if(itl != begin())
         if((--itl)->second < l) ++itl;
      if(itl == itr) return;
      int tl = min(l, itl->first), tr = max(r, prev(itr)->second);
      erase(itl, itr);
      if(tl < l) (*this)[tl] = l - 1;
      if(r < tr) (*this)[r + 1] = tr;
   }

   bool same(ll p, ll q) {
      auto it = get(p);
      return it != end() && it->first <= q && q <= it->second;
   }
};
```

range_tree.hpp md5: 7f74d5

```
template<class K, class M> struct range_tree {
   using S = typename M::S;
   using D = typename M::D;

   private:
   vector<pair<K, K>> ps;
   vector<K> xs;
   vector<vector<K>> ys;
   vector<D> ds;
   int n;
   int id(K x) const { return lower_bound(all(xs), x) - xs.begin();
}

   int id(int k, K y) const { return lower_bound(all(ys[k]), y) -
ys[k].begin(); }

   public:
   void add(K x, K y) { ps.emplace_back(x, y); }
   void build() {
      sort(ps.begin(), ps.end());
      ps.erase(unique(all(ps)), ps.end());
      n = size(ps);
      xs.reserve(n);
      for(auto& [x, _] : ps) xs.push_back(x);
      ys.resize(2 * n);
      ds.resize(2 * n, M::init(0));
      for(int i = 0; i < n; i++) {
         ys[i + n] = {ps[i].second};
         ds[i + n] = M::init(1);
      }
      for(int i = n - 1; i > 0; i--) {
```

```
         ys[i].resize(size(ys[i << 1]) + size(ys[(i << 1) | 1]));
         merge(all(ys[i << 1]), all(ys[(i << 1) | 1]),
ys[i].begin());
         ys[i].erase(unique(all(ys[i])), ys[i].end());
         ds[i] = M::init(size(ys[i]));
      }
   }

   void apply(K x, K y, S a) {
      int k = lower_bound(all(ps), make_pair(x, y)) - ps.begin() +
n;
      while(k > 0) {
         M::apply(ds[k], id(k, y), a);
         k >>= 1;
      }
   }

   S prod(K x1, K y1, K x2, K y2) {
      int a = id(x1), b = id(x2);
      a += n;
      b += n;
      S l = M::e(), r = M::e();
      while(a < b) {
         if(a & 1) {
            l = M::op(l, M::prod(ds[a], id(a, y1), id(a, y2)));
            ++a;
         }
         if(b & 1) {
            --b;
            r = M::op(M::prod(ds[b], id(b, y1), id(b, y2)), r);
         }
         a >>= 1;
         b >>= 1;
      }
      return M::op(l, r);
   }
};

/* 使い方

// モノイド
struct M {
   using S = ll;  // データ(モノイド)の型
   using D = BIT;  // ノードに持たせるデータ構造の型
   static S op(S a, S b) { return a + b; } // Sの二項演算
   static S e() { return 0; }  // Sの単位元
   static D init(int n) { return BIT(n); }  // Dを長さnで初期化する関
数
   static void apply(D& bit, int k, const S& v) { bit.add(k, v); }
// D のk番目に v を適用する関数
   static S prod(D& bit, int l, int r) { return bit.sum(l, r); } //
D の[l, r) に対するクエリを行う関数
};

rt.add(x, y): 座標 (x, y) を追加
rt.build(): クエリを受け付ける準備をする
rt.apply(x, y, a): 座標 (x, y) に a を適用
rt.prod(x1, y1, x2, y2): 座標 x \in [x1, x2), y \in [y1, y2) の領域に
クエリを行う
*/
```

segtree.hpp md5: d32488

```
// base: bafcf8
unsigned int bit_ceil(unsigned int n) {
   unsigned int x = 1;
   while(x < (unsigned int)(n)) x *= 2;
   return x;
}
int countr_zero(unsigned int n) { return __builtin_ctz(n); }
constexpr int countr_zero_constexpr(unsigned int n) {
   int x = 0;
   while(!(n & (1 << x))) x++;
   return x;
}
template<class S, S (*op)(S, S), S (*e)()> struct segtree {
   public:
   segtree() : segtree(0) {}
   explicit segtree(int n) : segtree(vector<S>(n, e())) {}
   explicit segtree(const vector<S>& v) : _n(int(v.size())) {
      size = (int)bit_ceil((unsigned int)(_n));
      log = countr_zero((unsigned int)size);
      d = vector<S>(2 * size, e());
      for(int i = 0; i < _n; i++) d[size + i] = v[i];
      for(int i = size - 1; i >= 1; i--) { update(i); }
   }

   void set(int p, S x) {
      // assert(0 <= p && p < _n);
      p += size;
      d[p] = x;
      for(int i = 1; i <= log; i++) update(p >> i);
   }

   S get(int p) const {
      // assert(0 <= p && p < _n);
      return d[p + size];
   }

   S prod(int l, int r) const {
      // assert(0 <= l && l <= r && r <= _n);
      S sml = e(), smr = e();
      l += size;
      r += size;

      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<class F> int max_right(int l, F f) {
      // assert(0 <= l && l <= _n);
      // assert(f(e()));
      if(l == _n) return _n;
      l += size;
      S sm = e();
      do {
         while(l % 2 == 0) l >>= 1;
         if(!f(op(sm, d[l]))) {
            while(l < size) {
               l = (2 * l);
               if(f(op(sm, d[l]))) {
                  sm = op(sm, d[l]);
                  l++;
               }
            }
            return l - size;
         }
         sm = op(sm, d[l]);
         l++;
      } while((l & -l) != l);
      return _n;
   }  // faa03f

   template<class F> int min_left(int r, F f) {
      // assert(0 <= r && r <= _n);
      // assert(f(e()));
      if(r == 0) return 0;
      r += size;
      S sm = e();
      do {
         r--;
         while(r > 1 && (r % 2)) r >>= 1;
         if(!f(op(d[r], sm))) {
            while(r < size) {
               r = (2 * r + 1);
               if(f(op(d[r], sm))) {
                  sm = op(d[r], sm);
                  r--;
               }
            }
            return r + 1 - size;
         }
         sm = op(d[r], sm);
      } while((r & -r) != r);
      return 0;
   }  // efa466

   private:
   int _n, size, log;
   vector<S> d;

   void update(int k) { d[k] = op(d[2 * k], d[2 * k + 1]); }
};
```

treap.hpp md5: fd1c1c

```
// base: c8a607
// mmを使う場合, 追記が必要
```

```
// friend bool operator==(const mm& a, const mm& b) { return a.x ==
b.x; }
template<class S, S (*op)(S, S), S (*e)(), class F, S (*mapping)(F,
S, int), F (*composition)(F, F), F (*id)()>
struct Treap {
   private:
   mt19937_64 mt;
   uniform_int_distribution<uint64_t> rand;
   vector<S> value, acc;
   vector<F> lazy;
   vector<ll> priority;
   vector<int> cnt, lch, rch;
   vector<bool> lazy_rev;
   int new_node(S v, ll p) {
      value.push_back(v);
      acc.push_back(e());
      lazy.push_back(id());
      priority.push_back(p);
      cnt.push_back(0);
      lazy_rev.push_back(false);
      lch.push_back(-1);
      rch.push_back(-1);
      return value.size() - 1;
   }

   int root = -1;
   int get_cnt(int t) { return t == -1 ? 0 : cnt[t]; }
   S get_acc(int t) { return t == -1 ? e() : acc[t]; }
   int update(int t) {
      if(t == -1) return t;
      cnt[t] = 1 + get_cnt(lch[t]) + get_cnt(rch[t]);
      acc[t] = op(get_acc(lch[t]), op(value[t], get_acc(rch[t])));
      return t;
   }
   int push(int t) {
      if(t == -1) return t;
      if(lazy_rev[t]) {
         lazy_rev[t] = false;
         swap(lch[t], rch[t]);
         if(lch[t] != -1) lazy_rev[lch[t]] = !lazy_rev[lch[t]];
         if(rch[t] != -1) lazy_rev[rch[t]] = !lazy_rev[rch[t]];
      }
      if(lazy[t] != id()) {
         if(lch[t] != -1) {
            lazy[lch[t]] = composition(lazy[t], lazy[lch[t]]);
            acc[lch[t]] = mapping(lazy[t], acc[lch[t]],
get_cnt(lch[t]));
         }
         if(rch[t] != -1) {
            lazy[rch[t]] = composition(lazy[t], lazy[rch[t]]);
            acc[rch[t]] = mapping(lazy[t], acc[rch[t]],
get_cnt(rch[t]));
         }
         value[t] = mapping(lazy[t], value[t], 1);
         lazy[t] = id();
      }
      return update(t);
   }

   int merge(int l, int r) {
      push(l);
      push(r);
      if(l == -1) return r;
      if(r == -1) return l;
      if(priority[l] > priority[r]) {
         rch[l] = merge(rch[l], r);
         return update(l);
      } else {
         lch[r] = merge(l, lch[r]);
         return update(r);
      }
   }
   pair<int, int> split(int t, int k) {
      if(t == -1) return make_pair(-1, -1);
      push(t);
      int implicit_key = get_cnt(lch[t]) + 1;
      if(k < implicit_key) {
         auto s = split(lch[t], k);
         lch[t] = s.second;
         return make_pair(s.first, update(t));
      } else {
         auto s = split(rch[t], k - implicit_key);
         rch[t] = s.first;
         return make_pair(update(t), s.second);
      }
   }

   int insert(int t, int k, int n) {
      auto s = split(t, k);
      return merge(merge(s.first, n), s.second);
   }

   int apply(int t, int l, int r, F f) {
      auto [t1, tt] = split(t, l);
      auto [t2, t3] = split(tt, r - l);
      lazy[t2] = composition(f, lazy[t2]);
      acc[t2] = mapping(f, acc[t2], get_cnt(t2));
      return merge(merge(t1, t2), t3);
   }  // 905a19 (Unordered)

   int _erase(int t, int k) {
      auto [tt, t3] = split(t, k + 1);
      auto [t1, t2] = split(tt, k);
      return merge(t1, t3);
   }  // 92ef20 (Common)

   int erase_range(int t, int l, int r) {
      auto [tt, t3] = split(t, r);
      auto [t1, t2] = split(tt, l);
      return merge(t1, t3);
   }  // 77074b (Common)

   pair<S, int> query(int t, int l, int r) {
      auto [t1, tt] = split(t, l);
      auto [t2, t3] = split(tt, r - l);
      S ret = acc[t2];
      return make_pair(ret, merge(merge(t1, t2), t3));
   }  //  fe8e6c (Common)

   int set(int t, int k, S v) {
      auto [tt, t3] = split(t, k + 1);
      auto [t1, t2] = split(tt, k);
      push(t2);
      value[t2] = v;
      update(t2);
      return merge(merge(t1, t2), t3);
   }  // 31b211 (Unordered)

   int _find(int t, S x, int offset, bool left = true) {
      if(op(get_acc(t), x) == x) {
         return -1;
      } else {
         if(left) {
            if(lch[t] != -1 && op(acc[lch[t]], x) != x) {
               return find(lch[t], x, offset, left);
            } else {
               return (op(value[t], x) != x) ? offset +
get_cnt(lch[t])
                                             : find(rch[t], x,
offset + get_cnt(lch[t]) + 1, left);
            }
         } else {
            if(rch[t] != -1 && op(acc[rch[t]], x) != x) {
               return find(rch[t], x, offset + get_cnt(lch[t]) + 1,
left);
            } else {
               return (op(value[t], x) != x) ? offset +
get_cnt(lch[t]) : find(lch[t], x, offset, left);
            }
         }
      }
   }  // b0c65b (Common)

   int reverse(int t, int l, int r) {
      auto [t1, tt] = split(t, l);
      auto [t2, t3] = split(tt, r - l);
      lazy_rev[t2] = !lazy_rev[t2];
      return merge(merge(t1, t2), t3);
   }  // 3f67e3 (Unordered)

   int rotate(int t, int l, int m, int r) {
      t = reverse(t, l, r);
      t = reverse(t, l, l + r - m);
      return reverse(t, l + r - m, r);
   }  // a5a67c (Unordered)

   int lower_search(int t, S x) {
      int ret = 0;
      while(t != -1) {
         if(x <= value[t]) {
            t = lch[t];
         } else {
            ret += get_cnt(lch[t]) + 1;
            t = rch[t];
         }
      }
```

```
      return ret;
   }  // 0ef7d9 (Ordered)

   int upper_search(int t, S x) {
      int ret = 0;
      while(t != -1) {
         if(x < value[t]) {
            t = lch[t];
         } else {
            ret += get_cnt(lch[t]) + 1;
            t = rch[t];
         }
      }
      return ret;
   }  // f91898 (Ordered)

   public:
   Treap() : Treap(0) {}
   Treap(int N) : Treap(vector<S>(N, e())) {}
   Treap(vector<S> V) {
      mt =
mt19937_64(chrono::steady_clock::now().time_since_epoch().count());
      rand = uniform_int_distribution<uint64_t>(1, 1e18);
      for(auto v : V) { push_back(v); }
   }

   size_t size() { return size_t(get_cnt(root)); }
   // f63788 (Common)

   void insert(int ind, S x) { root = insert(root, ind, new_node(x,
rand(mt))); }
   // dc467c (UnOrdered)

   void push_back(S x) { root = insert(root, int(size()),
new_node(x, rand(mt))); }
   // 7fa616 (Unordered)

   void ordered_insert(S x) {
      int ind = lower_search(root, x);
      insert(ind, x);
   }  // 539d77 (Ordered)

   // Count elements in [lower, upper)
   int value_range_cnt(S lower, S upper) {
      int L = lower_search(root, lower);
      int R = lower_search(root, upper);
      return R - L;
   }  // 2d4406 (Ordered)

   // Sum of elements in [lower, upper)
   S value_range_prod(S lower, S upper) {
      int L = lower_search(root, lower);
      int R = lower_search(root, upper);
      if(L == R) return e();
      return query(L, R);
   }  // 27b9d4 (Ordered)

   // erase element x cnt times (cnt = -1 -> erase all x)
   int erase_value(S x, int cnt = -1) {
      int L = lower_search(root, x);
      int R = upper_search(root, x);
      if(cnt != -1) chmin(R, L + cnt);
      root = erase_range(root, L, R);
      return R - L;
   }  // 5c60fd (Ordered)

   int lower_search(S x) { return lower_search(root, x); }
   // 9731cc (Ordered)

   int upper_search(S x) { return upper_search(root, x); }
   // ac5aa0 (Ordered)

   void apply(int l, int r, F f) { root = apply(root, l, r, f); }
   // 905a19 (Unordered)

   void erase(int ind) { root = _erase(root, ind); }
   // ff257f (Common)

   void erase(int l, int r) {
      auto [tt, t3] = split(root, r);
      auto [t1, t2] = split(tt, l);
      root = merge(t1, t3);
   }
   // f9ff4a (Common)

   // l .. r-1 -> r-1 .. l
   void reverse(int l, int r) { root = reverse(root, l, r); }
   // 40df7d (Unordered)

   // l .. m-1, m .. r-1 -> m .. r-1, l .. m-1
   void rotate(int l, int m, int r) { root = rotate(root, l, m, r);
}
   // e21b85 (Unordered)

   void set(int k, S v) { root = set(root, k, v); }
   // 4ae943 (Unordered)

   // min k \in [l,r) such that op(tr[k], x) != x
   int find(int l, int r, S x, bool left = true) {
      auto [t1, tt] = split(root, l);
      auto [t2, t3] = split(tt, r - l);
      int ret = _find(t2, x, l, left);
      if(ret == -1) ret = r;
      root = merge(merge(t1, t2), t3);
      return ret;
   }  // 4f1699 (Common)

   S prod(int l, int r) {
      if(l == r) return S(0);
      auto [t, rt] = query(root, l, r);
      root = rt;
      return t;
   }  // c46ac4 (Common)

   S operator[](int ind) {
      auto [tt, t3] = split(root, ind + 1);
      auto [t1, t2] = split(tt, ind);
      S ret = acc[t2];
      root = merge(merge(t1, t2), t3);
      return ret;
   }  // d2546e (Common)
};
```

undo_dsu.hpp md5: f5d93b

```
// base: edf246
struct undo_dsu {
   private:
   int _n;
   vector<int> p;
   stack<pair<int, int>> history;

   public:
   undo_dsu() : _n(0) {}
   explicit undo_dsu(int n) : _n(n), p(n, -1) {}

   int merge(int a, int b) {
      // assert(0 <= a && a < _n);
      // assert(0 <= b && b < _n);
      int x = leader(a), y = leader(b);
      if(x == y) {
         history.emplace(x, p[x]);
         history.emplace(y, p[y]);
         return x;
      }
      if(-p[x] < -p[y]) swap(x, y);
      history.emplace(x, p[x]);
      history.emplace(y, p[y]);
      p[x] += p[y];
      p[y] = x;
      return x;
   }

   bool same(int a, int b) {
      // assert(0 <= a && a < _n);
      // assert(0 <= b && b < _n);
      return leader(a) == leader(b);
   }

   int leader(int a) {
      // assert(0 <= a && a < _n);
      while(p[a] >= 0) a = p[a];
      return a;
   }

   void undo() {
      p[history.top().first] = history.top().second;
      history.pop();
      p[history.top().first] = history.top().second;
      history.pop();
   }

   int snapshot() { return history.size(); }

   void rollback(int snapshot = 0) {
      while(history.size() > snapshot) undo();
   }
```

```
   int size(int a) {
      // assert(0 <= a && a < _n);
      return -p[leader(a)];
   }  // 818fe7
};
```

wavelet_matrix.hpp md5: 208fa9

```
// base: be292e
struct BitVector {
   private:
   vector<int> vec;

   public:
   BitVector(const vector<int>& a) {
      vec.resize(a.size() + 1);
      for(int j = 0; j < (int)a.size(); j++) { vec[j + 1] = vec[j]
+ a[j]; }
   }

   int get(const int i) { return vec[i + 1] - vec[i]; }

   int rank(const int b, const int i) {
      if(b == 0) return i - vec[i];
      else return vec[i];
   }
};

template<typename T, int bit_len = 62> struct WaveletMatrix {
   private:
   vector<BitVector> B;
   vector<vector<T>> acc;
   vector<int> so;
   map<T, int> sn;
   int len;

   public:
   WaveletMatrix(vector<T> vec) {
      len = vec.size();
      acc = vector<vector<T>>(bit_len, vector<T>(len + 1));
      so = vector<int>(bit_len);
      vector<T> v(vec);
      for(int b = 0; b < bit_len; b++) {
         vector<T> cur;
         cur.reserve(len);
         vector<int> bi(len);
         for(int i = 0; i < len; i++) {
            ll bit = (v[i] >> (bit_len - b - 1)) & 1;
            if(bit == 0) {
               cur.push_back(v[i]);
               bi[i] = 0;
            }
         }
         so[b] = cur.size();
         for(int i = 0; i < len; i++) {
            ll bit = (v[i] >> (bit_len - b - 1)) & 1;
            if(bit == 1) {
               cur.push_back(v[i]);
               bi[i] = 1;
            }
         }
         B.push_back(BitVector(bi));
         for(int i = 0; i < len; i++) {
            if(B[b].get(i) == 0) acc[b][i + 1] = v[i];
            acc[b][i + 1] += acc[b][i];
         }
         v = cur;
      }
      for(int i = len - 1; i >= 0; i--) { sn[v[i]] = i; }
   }

   T access(int i) {
      T ret = 0;
      for(int j = 0; j < bit_len; j++) {
         int bit = B[j].get(i);
         ret = (ret << 1) | bit;
         i = B[j].rank(bit, i) + so[j] * bit;
      }
      return ret;
   }  // 3be264

   int rank(T val, int i) {
      if(!sn.count(val)) return 0;
      for(int j = 0; j < bit_len; j++) {
         int bit = (val >> (bit_len - j - 1)) & 1;
         i = B[j].rank(bit, i) + so[j] * bit;
      }
      return i - sn[val];
   }  // 88f41a

   T kthMin(int left, int right, int k) {
      T ret = 0;
      for(int j = 0; j < bit_len; j++) {
         int l = B[j].rank(0, left);
         int r = B[j].rank(0, right);
         int cnt = r - l;
         if(cnt > k) {
            left = l;
            right = r;
         } else {
            k -= cnt;
            left += so[j] - l;
            right += so[j] - r;
            ret |= (1LL << (bit_len - j - 1));
         }
      }
      return ret;
   }  // 941aa0

   T kMinSum(int left, int right, int k) {
      T ret = 0;
      for(int j = 0; j < bit_len; j++) {
         int l = B[j].rank(0, left);
         int r = B[j].rank(0, right);
         int cnt = r - l;
         if(cnt > k) {
            left = l;
            right = r;
         } else {
            k -= cnt;
            ret += acc[j][right] - acc[j][left];
            left += so[j] - l;
            right += so[j] - r;
         }
      }
      return ret;
   }  // edb4f5

   int lessCount(int left, int right, T upper) {
      int ret = 0;
      if(upper >= (1LL << bit_len)) return right - left;
      for(int j = 0; j < bit_len; j++) {
         int l = B[j].rank(0, left);
         int r = B[j].rank(0, right);
         int cnt = r - l;
         if((upper >> (bit_len - j - 1)) & 1) {
            ret += cnt;
            left += so[j] - l;
            right += so[j] - r;
         } else {
            left = l;
            right = r;
         }
      }
      return ret;
   }  // 029c6d
};
```

math

BinaryGCD.hpp md5: f3ab31

```
u64 ctz(u64 x) { return countr_zero(x); }
u64 binary_gcd(u64 x, u64 y) {
   if(!x || !y) return x | y;
   u64 n = ctz(x), m = ctz(y);
   x >>= n, y >>= m;
   while(x != y) {
      if(x > y) x = (x - y) >> ctz(x - y);
      else y = (y - x) >> ctz(y - x);
   }
   return x << min(n, m);
}
```

ExtGCD.hpp md5: c3fa9b

```
// 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;
}
```

combination.hpp md5: 5a309a

```cpp
struct Combination {
   ll C_MOD;
   vector<ll> fac, finv, inv;
   Combination(ll n, ll mod) : C_MOD(mod) {
      n = max(n, 2LL);
      fac.resize(n, 0);
      finv.resize(n, 0);
      inv.resize(n, 0);
      fac[0] = fac[1] = finv[0] = finv[1] = inv[1] = 1;
      for(int i = 2; i < n; i++) {
         fac[i] = fac[i - 1] * i % mod;
         inv[i] = mod - inv[mod % i] * (mod / i) % mod;
         finv[i] = finv[i - 1] * inv[i] % mod;
      }
   }
   ll com(ll n, ll k) {
      if(n < k || n < 0 || k < 0) return 0;
      return fac[n] * (finv[k] * finv[n - k] % C_MOD) % C_MOD;
   }
};
```

crt.hpp md5: 338cb4

```cpp
// mが互いに素になるように前処理
// lcm(m) % MODを返す
// crtの解がなければ-1を返す
ll pre_garner(vector<ll>& b, vector<ll>& m, ll MOD) {
   ll res = 1;
   for(int i = 0; i < (int)b.size(); i++) {
      for(int j = 0; j < i; j++) {
         ll g = gcd(m[i], m[j]);
         if((b[i] - b[j]) % g != 0) return -1;
         m[i] /= g;
         m[j] /= g;
         ll gi = gcd(m[i], g);
         ll gj = g / gi;
         do {
            g = gcd(gi, gj);
            gi *= g;
            gj /= g;
         } while(g != 1);
         m[i] *= gi;
         m[j] *= gj;
         b[i] %= m[i];
         b[j] %= m[j];
      }
   }
   for(int i = 0; i < size(b); i++) (res *= m[i]) %= MOD;
   return res;
}

// m が互いに素であることが保証されている場合
// b[i] = x (mod m[i]) となる最小の x \le 0 を求める
ll garner(vector<ll> b, vector<ll> m, ll MOD) {
   m.push_back(MOD);
   vector<ll> coeffs(size(m), 1);
   vector<ll> constants(size(m), 0);
   for(int k = 0; k < size(b); k++) {
      ll t = ((b[k] - constants[k]) * modinv(coeffs[k], m[k])) %
m[k];
      if(t < 0) t += m[k];
      for(int i = k + 1; i < size(m); i++) {
         (constants[i] += t * coeffs[i]) %= m[i];
         (coeffs[i] *= m[k]) %= m[i];
      }
   }
   return constants.back();
}
```

floor_sum.hpp md5: 0f7242

```cpp
ll floor_sum(const ll& n, const ll& m, ll a, ll b) {
   ll ret = 0;
   if(a >= m) ret += (n - 1) * n * (a / m) / 2, a %= m;
   if(b >= m) ret += n * (b / m), b %= m;
   ll y = (a * n + b) / m;
   if(y == 0) return ret;
   ll x = y * m - b;
   ret += (n - (x + a - 1) / a) * y;
   ret += floor_sum(y, a, m, (a - x % a) % a);
   return ret;
}
```

fwt.hpp md5: f1dce9

```cpp
template<typename T> void fwt(vector<T>& f) {
   int n = f.size();
   for(int i = 1; i < n; i <<= 1) {
      for(int j = 0; j < n; j++) {
         if((j & i) == 0) {
            T x = f[j], y = f[j | i];
            f[j] = x + y, f[j | i] = x - y;
         }
      }
   }
}
template<typename T> void ifwt(vector<T>& f) {
   int n = f.size();
   for(int i = 1; i < n; i <<= 1) {
      for(int j = 0; j < n; j++) {
         if((j & i) == 0) {
            T x = f[j], y = f[j | i];
            f[j] = (x + y) / 2, f[j | i] = (x - y) / 2;
         }
      }
   }
}
```

lagrange_polynomial.hpp md5: 5e6e39

```cpp
template<typename T> T lagrange_polynomial(const vector<T>& y, ll
t, ll mod = 1000000007) {
   int n = y.size() - 1;
   if(t <= n) return y[t];
   T ret(0);
   Combination comb(n + 1, mod);
   vector<T> dp(n + 1, 1), pd(n + 1, 1);
   for(int i = 0; i < n; i++) dp[i + 1] = dp[i] * (t - i);
   for(int i = n; i > 0; i--) pd[i - 1] = pd[i] * (t - i);
   for(int i = 0; i <= n; i++) {
      T tmp = y[i] * dp[i] * pd[i] * comb.finv[i] * comb.finv[n -
i];
      ret -= ((n - i) & 1 ? tmp : T(0) - tmp);
   }
   return ret;
}
```

min_of_mod_of_linear.hpp md5: 30d270

```cpp
// depends on floor_sum
ll min_of_mod_of_linear(ll n, ll m, ll a, ll b) {
   ll fsum = floor_sum(n, m, a, b);
   ll le = -1, ri = m - 1;
   while(ri - le > 1) {
      ll mid = (le + ri) / 2;
      if(floor_sum(n, m, a, b + (m - 1 - mid)) < fsum + n) ri =
mid;
      else le = mid;
   }
   return ri;
}
```

modinv.hpp md5: a0de19

```cpp
ll modinv(ll a, ll MOD) {
   ll b = MOD, u = 1, v = 0;
   while(b) {
      ll t = a / b;
      a -= t * b;
      swap(a, b);
      u -= t * v;
      swap(u, v);
   }
   u %= MOD;
   if(u < 0) u += MOD;
   return u;
}
```

modlog.hpp md5: 74b856

```cpp
// depends on modpow and modinv
// a^x ≡ b (mod. m) となる最小の正の整数 x を求める
long long modlog(long long a, long long b, int m) {
   a %= m, b %= m;

   // calc sqrt{M}
   long long le = -1, ri = m;
   while(ri - le > 1) {
      long long mid = (le + ri) >> 1;
      if(mid * mid >= m) ri = mid;
      else le = mid;
   }
```

```
   long long sqrtM = ri;

   // {a^0, a^1, a^2, ..., a^sqrt(m)}
   map<long long, long long> apow;
   long long r = a;
   for(long long i = 1; i < sqrtM; ++i) {
      if(!apow.count(r)) apow[r] = i;
      (r *= a) %= m;
   }

   // check each A^p
   long long A = modpow(modinv(a, m), sqrtM, m);
   r = b;
   for(long long q = 0; q < sqrtM; ++q) {
      if(r == 1 && q > 0) return q * sqrtM;
      else if(apow.count(r)) return q * sqrtM + apow[r];
      (r *= A) %= m;
   }

   // no solutions
   return -1;
}
```

modsqrt.hpp md5: bf9df0

```
// depends on modpow
// a is a quadratic residue modulo p?
int Legendre(long long a, long long p) {
   long long ret = modpow(a, (p - 1) / 2, p);

   if(ret == p - 1) ret = -1;

   return ret;
}

// Tonelli-Shanks algorithm
// calculate R such that R^2 = n (mod p)
// p is an odd prime
// n is a positive integer that satisfies (n/p) = 1
// See
http://en.wikipedia.org/wiki/Tonelli%E2%80%93Shanks_algorithm
// O((log p)^2)
long long modsqrt(long long n, long long p) {
   // step 1
   long long Q = p - 1;
   long long S = 0;
   while(Q % 2 == 0) {
      ++S;
      Q /= 2;
   }

   if(S == 1) { return modpow(n, (p + 1) / 4, p); }

   // step 2
   default_random_engine generator;
   uniform_int_distribution<long long> distribution(0, p);

   long long z = 1;
   while(Legendre(z, p) != -1) { z = distribution(generator); }

   long long c = modpow(z, Q, p);

   // step 3
   long long R = modpow(n, (Q + 1) / 2, p);
   long long t = modpow(n, Q, p);
   long long M = S;

   // step 4
   while(t != 1) {
      long long i;
      long long t2 = t;
      for(i = 1; i < M; i++) {
         t2 = t2 * t2 % p;
         if(t2 == 1) break;
      }

      long long b = modpow(c, 1LL << (M - i - 1), p);
      R = R * b % p;
      t = (t * b % p) * b % p;
      c = b * b % p;
      M = i;
   }

   return R;
}
```

primality.hpp md5: d6eb6a

```
bool is_prime(ll N) {
   if(N == 2) return true;
   if(N == 1 || N % 2 == 0) return false;
   ll s = 0;
   ll d = N - 1;
   while(d % 2 == 0) {
      s++;
      d /= 2;
   }
   vector<ll> tests = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
   for(auto a : tests) {
      if(a == N) continue;
      ll X = modpow(a, d, N);
      int r = 0;
      if(X == 1) { continue; }
      while(X != N - 1) {
         X = modpow(X, 2, N);
         r++;
         if(X == 1 || r == s) return false;
      }
   }
   return true;
}
```

primitive-root.hpp md5: a7c93d

```
// depends on modpow
int primitive_root(int m) {
   if(m == 2) return 1;
   if(m == 167772161) return 3;
   if(m == 469762049) return 3;
   if(m == 754974721) return 11;
   if(m == 998244353) return 3;
   int divs[20] = {};
   divs[0] = 2;
   int cnt = 1;
   int x = (m - 1) / 2;
   while(x % 2 == 0) x /= 2;
   for(int i = 3; (long long)(i)*i <= x; i += 2) {
      if(x % i == 0) {
         divs[cnt++] = i;
         while(x % i == 0) { x /= i; }
      }
   }
   if(x > 1) { divs[cnt++] = x; }
   for(int g = 2;; g++) {
      bool ok = true;
      for(int i = 0; i < cnt; i++) {
         if(modpow(g, (m - 1) / divs[i], m) == 1) {
            ok = false;
            break;
         }
      }
      if(ok) return g;
   }
}
```

rho.hpp md5: 820144

```
ll find_prime_factor(ll N) {
   using i128 = __int128_t;
   if(N % 2 == 0) { return 2; }
   int b = int(sqrt(sqrt(sqrt(N)))) + 1;
   for(ll c = 1; c < N; c++) {
      auto f = [&](ll a) -> ll { return modpow(a, 2, N) + c; };
      ll y = 6;
      ll g = 1;
      i128 q = 1;
      int r = 1;
      int k = 0;
      ll ys = 0;
      ll x = 0;
      while(g == 1) {
         x = y;
         while(k < 3 * r / 4) {
            y = f(y);
            k++;
         }
         while(k < r && g == 1) {
            ys = y;
            for(ll i = 0; i < min(b, r - k); i++) {
               y = f(y);
               q *= abs(x - y);
               q %= N;
            }
            g = gcd(ll(q), N);
```

```
            k += b;
        }
        k = r;
        r *= 2;
    }
    if(g == N) {
        g = 1;
        y = ys;
        while(g == 1) {
            y = f(y);
            g = gcd(abs(x - y), N);
        }
    }
    if(g == N) { continue; }
    if(is_prime(g)) { return g; }
    if(is_prime(N / g)) { return N / g; }
    return find_prime_factor(g);
  }
  assert(false);
}

map<ll, int> factorize(ll N) {
  map<ll, int> ret;
  while(!is_prime(N) && N > 1) {
    ll p = find_prime_factor(N);
    int s = 0;
    while(N % p == 0) {
      N /= p;
      s++;
    }
    ret[p] = s;
  }
  if(N > 1) { ret[N] = 1; }
  return ret;
}
```

modint

BarrettReduction.hpp

md5: 2ca7f3

```
// using u64 = uint64_t;
struct Barrett {  // mod < 2^32
  u64 m, im;
  Barrett(u64 mod) : m(mod), im(-1ULL / m + 1) {}
  // input: a * b < 2^64, output: a * b % mod
  u64 mul(u64 a, u64 b) const {
    a *= b;
    u64 x = ((__uint128_t)a * im) >> 64;
    a -= x * m;
    if((ll)a < 0) a += m;
    return a;
  }
};
```

modint.hpp

md5: eb2b53

```
const ll mod = 998244353;
struct mm {
  ll x;
  mm(ll x_ = 0) : x(x_ % mod) {
    if(x < 0) x += mod;
  }
  friend mm operator-(mm a) { return -a.x; }
  friend mm operator+(mm a, mm b) { return a.x + b.x; }
  friend mm operator-(mm a, mm b) { return a.x - b.x; }
  friend mm operator*(mm a, mm b) { return a.x * b.x; }
  friend mm operator/(mm a, mm b) { return a * b.inv(); }
  // 4 行コピペ  Alt + Shift + クリックで複数カーソル
  friend mm& operator+=(mm& a, mm b) { return a = a.x + b.x; }
  friend mm& operator-=(mm& a, mm b) { return a = a.x - b.x; }
  friend mm& operator*=(mm& a, mm b) { return a = a.x * b.x; }
  friend mm& operator/=(mm& a, mm b) { return a = a * b.inv(); }
  mm inv() const { return pow(mod - 2); }
  mm pow(ll b) const {
    mm a = *this, c = 1;
    while(b) {
      if(b & 1) c *= a;
      a *= a;
      b >>= 1;
    }
    return c;
  }
};
```

FPS

FFT.hpp

md5: 6e60c3

```
// {998244353, 3}, {1811939329, 13}, {2013265921, 31}
mm g = 3;  // 原始根
void fft(vector<mm>& a) {
  ll n = size(a), lg = __lg(n);
  assert((1 << lg) == n);
  vector<mm> b(n);
  for(int l = 1; l <= lg; l++) {
    ll w = n >> l;
    mm s = 1, r = g.pow(mod >> l);
    for(ll u = 0; u < n / 2; u += w) {
      for(int d = 0; d < w; d++) {
        mm x = a[u << 1 | d], y = a[u << 1 | w | d] * s;
        b[u | d] = x + y;
        b[n >> 1 | u | d] = x - y;
      }
      s *= r;
    }
    swap(a, b);
  }
}
vector<mm> conv(vector<mm> a, vector<mm> b) {
  if(a.empty() || b.empty()) return {};
  size_t s = size(a) + size(b) - 1, n = bit_ceil(s);
  // if(min(sz(a), sz(b)) <= 60) 愚直に掛け算
  a.resize(n);
  b.resize(n);
  fft(a);
  fft(b);
  mm inv = mm(n).inv();
  for(int i = 0; i < n; i++) a[i] *= b[i] * inv;
  reverse(1 + all(a));
  fft(a);
  a.resize(s);
  return a;
}
```

barlekamp_massey.hpp

md5: 50177f

```
vector<mm> BerlekampMassey(const vector<mm>& s) {
  const int N = (int)s.size();
  vector<mm> b, c;
  b.reserve(N + 1);
  c.reserve(N + 1);
  b.push_back(mm(1));
  c.push_back(mm(1));
  mm y = mm(1);
  for(int ed = 1; ed <= N; ed++) {
    int l = int(c.size()), m = int(b.size());
    mm x = 0;
    for(int i = 0; i < l; i++) x += c[i] * s[ed - l + i];
    b.emplace_back(mm(0));
    m++;
    if(x.x == 0) continue;
    mm freq = x / y;
    if(l < m) {
      auto tmp = c;
      c.insert(begin(c), m - l, mm(0));
      for(int i = 0; i < m; i++) c[m - 1 - i] -= freq * b[m - 1
- i];
      b = tmp;
      y = x;
    } else {
      for(int i = 0; i < m; i++) c[l - 1 - i] -= freq * b[m - 1
- i];
    }
  }
  reverse(begin(c), end(c));
  return c;
}
```

bostan_mori.hpp

md5: dbb7af

```
// find [x^N] P(x)/Q(x), O(K log K log N)
// deg(Q(x)) = K, deg(P(x)) < K, Q[0] = 1
mm BostanMori(vector<mm> P, vector<mm> Q, ll N) {
  const int d = Q.size();
  for(; N; N >>= 1) {
    auto Q_neg = Q;
    for(size_t i = 1; i < Q.size(); i += 2) Q_neg[i] *= -1;
    P = conv(P, Q_neg);
    Q = conv(Q, Q_neg);
    for(size_t i = N & 1; i < P.size(); i += 2) P[i >> 1] = P[i];
    for(size_t i = 0; i < Q.size(); i += 2) Q[i >> 1] = Q[i];
    P.resize(d - 1);
```

```
      Q.resize(d);
   }
   return P[0];
}
```

fps.hpp md5: 930ad2

```
using vm = vector<mm>;
vm pre(vm f, int sz) { return vm(f.begin(), f.begin() +
min((int)f.size(), sz)); }
vm inv(vm f, int deg = -1) {
   if(deg < 0) deg = (int)f.size();
   vm res({mm(1) / f[0]});
   for(int i = 1; i < deg; i <<= 1) {
      vm ff = conv(res, conv(res, pre(f, i << 1)));
      int sz = res.size();
      for(int j = 0; j < sz; j++) res[j] += res[j];
      sz = max(sz, (int)ff.size());
      res.resize(sz);
      sz = ff.size();
      for(int j = 0; j < sz; j++) res[j] -= ff[j];
      res = pre(res, i << 1);
   }
   res.resize(deg);
   return res;
}
vm diff(vm f) {
   int n = f.size();
   vm res(n - 1);
   for(int i = 1; i < n; i++) res[i - 1] = f[i] * i;
   return res;
}
vm intg(vm f) {
   int n = (int)f.size();
   vm res(n + 1, 0);
   for(int i = 0; i < n; i++) res[i + 1] = f[i] / (i + 1);
   return res;
}
vm log(vm f, int deg = -1) {
   if(deg == -1) deg = f.size();
   vm res = intg(conv(diff(f), inv(f, deg)));
   res.resize(deg);
   return res;
}
vm exp(vm f, int deg = -1) {
   vm res(1, 1);
   if(deg == -1) deg = f.size();
   for(int i = 1; i < deg; i <<= 1) {
      vm ff1 = pre(f, i << 1);
      vm ff2 = log(res, i << 1);
      ff1.resize(max(ff1.size(), ff2.size()));
      ff1[0] += mm(1);
      int sz = ff2.size();
      for(int j = 0; j < sz; j++) ff1[j] -= ff2[j];
      res = conv(res, pre(ff1, i << 1));
   }
   res.resize(deg);
   return res;
}
vm taylor_shift(vm f, mm a) {
   int n = f.size();
   vm fac(n, 1), inv(n, 1), finv(n, 1);
   for(int i = 2; i < n; i++) {
      fac[i] = fac[i - 1] * i;
      inv[i] = -inv[mod % i] * (mod / i);
      finv[i] = finv[i - 1] * inv[i];
   }
   for(int i = 0; i < n; i++) f[i] *= fac[i];
   std::reverse(f.begin(), f.end());
   vm g(n, 1);
   for(int i = 1; i < n; i++) g[i] = g[i - 1] * a * inv[i];
   f = pre(conv(f, g), n);
   reverse(f.begin(), f.end());
   for(int i = 0; i < n; i++) f[i] *= finv[i];
   return f;
}
```

fps_sparse.hpp md5: a351bf

```
using vm = vector<mm>;
vm fps_inv_sparse(vm& f) {
   int n = f.size();
   vector<pair<int, mm>> dat;
   for(int i = 0; i < n; i++)
      if(f[i].x) dat.emplace_back(i, f[i]);
   vm g(n);
   mm g0 = f[0].inv();
   g[0] = g0;
   for(int i = 1; i < n; i++) {
      mm rhs = 0;
      for(auto&& [k, fk] : dat) {
         if(k > i) break;
         rhs -= fk * g[i - k];
      }
      g[i] = rhs * g0;
   }
   return g;
}
vm fps_exp_sparse(vm& f) {
   if((int)f.size() == 0) return {mm(1)};
   int N = f.size();
   vector<pair<int, mm>> dat;
   for(int i = 1; i < N; i++)
      if(f[i].x) dat.emplace_back(i - 1, mm(i) * f[i]);
   vm F(N);
   F[0] = 1;
   for(int n = 1; n < N; n++) {
      mm rhs = 0;
      for(auto&& [k, fk] : dat) {
         if(k > n - 1) break;
         rhs += fk * F[n - 1 - k];
      }
      F[n] = rhs * mm(n).inv();
   }
   return F;
}
vm fps_log_sparse(vm& f) {
   int N = f.size();
   vector<pair<int, mm>> dat;
   for(int i = 1; i < N; i++)
      if(f[i].x) dat.emplace_back(i, f[i]);
   vm F(N), g(N - 1);
   for(int n = 0; n < N - 1; n++) {
      mm rhs = mm(n + 1) * f[n + 1];
      for(auto&& [i, fi] : dat) {
         if(i > n) break;
         rhs -= fi * g[n - i];
      }
      g[n] = rhs;
      F[n + 1] = rhs * mm(n + 1).inv();
   }
   return F;
}
vm fps_pow_product(vm& f, vm& g, mm n, mm m) {
   int N = f.size();
   using P = pair<int, mm>;
   vector<P> dat_f, dat_g;
   for(int i = 0; i < (int)f.size(); i++)
      if(f[i].x) dat_f.emplace_back(i, f[i]);
   for(int i = 0; i < (int)g.size(); i++)
      if(g[i].x) dat_g.emplace_back(i, g[i]);
   vm a(N), b(N);
   for(auto&& [i, x] : dat_f)
      for(auto&& [j, y] : dat_g) {
         if(i + j >= N + 1) continue;
         mm xy = x * y;
         if(i + j < N) a[i + j] += xy;
         if(0 < i + j && i + j <= N) b[i + j - 1] -= xy * (n *
mm(i) + m * mm(j));
      }
   vector<P> dat_a, dat_b;
   for(int i = 1; i < N; i++)
      if(a[i].x) dat_a.emplace_back(i, a[i]);
   for(int i = 0; i < N; i++)
      if(b[i].x) dat_b.emplace_back(i, b[i]);
   vm F(N), df(N - 1);
   F[0] = 1;
   for(int n = 0; n < N - 1; n++) {
      mm v = 0;
      for(auto&& [i, ai] : dat_a) {
         if(i > n) break;
         v += ai * df[n - i];
      }
      for(auto&& [i, bi] : dat_b) {
         if(i > n) break;
         v += bi * F[n - i];
      }
      df[n] = -v;
      F[n + 1] = df[n] * mm(n + 1).inv();
   }
   return F;
}
```

relaxed_conv.hpp md5: 33c76b

```
template<typename T> class RelaxedConvolution {
   int N, pos;
```

```
   vector<T> f, g, buf;
   vector<vector<tuple<int, int, int, int>>> event;
   void dfs1(int le, int ri) {
      if(ri - le == 1) {
         event[le].push_back({le, le + 1, 0, 1});
         return;
      }
      int mid = (le + ri) / 2;
      event[mid].push_back({le, mid, mid - le, ri - le});
      event[ri].push_back({mid, ri, mid - le, ri - le});
      dfs1(le, mid);
      dfs1(mid, ri);
   }
   void dfs2(int le, int ri) {
      if(ri - le == 1) {
         event[le].push_back({0, 1, le, le + 1});
         return;
      }
      int mid = (le + ri) / 2;
      event[mid].push_back({mid - le, ri - le, le, mid});
      event[ri].push_back({mid - le, ri - le, mid, ri});
      dfs2(le, mid);
      dfs2(mid, ri);
   }
   void dfs(int len) {
      if(len == 1) {
         event[0].push_back({0, 1, 0, 1});
         return;
      }
      int mid = len / 2;
      event[len].push_back({mid, len, mid, len});
      dfs(mid);
      dfs1(mid, len);
      dfs2(mid, len);
   }

   public:
   RelaxedConvolution(int n) {
      N = 1, pos = 0;
      while(N < n) N *= 2;
      f.resize(N);
      g.resize(N);
      buf.resize(N);
      event.resize(N + 1);
      dfs(N);
   }
   T get(T x, T y) {
      f[pos] = x, g[pos] = y;
      for(auto [fl, fr, gl, gr] : event[pos]) {
         vector<T> A({f.begin() + fl, f.begin() + fr});
         vector<T> B({g.begin() + gl, g.begin() + gr});
         auto ret = conv(A, B);
         int sz = ret.size();
         for(int i = 0; i < sz; i++) {
            if(i + fl + gl >= N) break;
            buf[i + fl + gl] += ret[i];
         }
      }
      return buf[pos++];
   }
};
```

graph

bi_connected_components.hpp

md5: 9883af

```
struct BiConnectedComponents : LowLink {
   public:
   using LowLink::bridge;
   using LowLink::g;
   using LowLink::low;
   using LowLink::ord;

   vector<int> comp;
   vector<vector<int>> tree;
   vector<vector<int>> group;

   void build(const vector<vector<int>>& g) {
      comp.assign(size(g), -1);
      int k = 0;
      for(int i = 0; i < size(comp); i++) {
         if(comp[i] == -1) { dfs(i, -1, k); }
      }
      group.resize(k);
      for(int i = 0; i < size(g); i++) {
group[comp[i]].push_back(i); }
      tree.resize(k);
      for(auto& e : bridge) {
         tree[comp[e.first]].push_back(comp[e.second]);
         tree[comp[e.second]].push_back(comp[e.first]);
      }
   }

   explicit BiConnectedComponents(const vector<vector<int>>& g) :
LowLink(g) { build(g); }

   private:
   vector<int> used;
   vector<pair<int, int>> tmp;

   void dfs(int cur, int pre, int& k) {
      if(pre != -1 && ord[pre] >= low[cur]) comp[cur] = comp[pre];
      else comp[cur] = k++;
      for(auto to : g[cur]) {
         if(comp[to] == -1) dfs(to, cur, k);
      }
   }
};
```

eulerian_trail.hpp

md5: 89bed1

```
// base: 72bf84
template<bool directed> struct EulerianTrail {
   vector<vector<pair<int, int>>> G;
   vector<pair<int, int>> es;
   int M;
   vector<int> usedV, usedE, deg;

   EulerianTrail(int N) : G(N), deg(N), usedV(N), M(0) {}

   void add_edge(int a, int b) {
      es.emplace_back(a, b);
      G[a].emplace_back(b, M);
      if(directed) {
         deg[a]++;
         deg[b]--;
      } else {
         G[b].emplace_back(a, M);
         deg[a]++;
         deg[b]++;
      }
      M++;
   }

   vector<int> go(int s) {
      stack<pair<int, int>> st;
      vector<int> ord;
      st.emplace(s, -1);
      while(!st.empty()) {
         int i = st.top().first;
         usedV[i] = true;
         if(G[i].empty()) {
            ord.emplace_back(st.top().second);
            st.pop();
         } else {
            auto e = G[i].back();
            G[i].pop_back();
            if(usedE[e.second]) continue;
            usedE[e.second] = true;
            if(!directed && es[e.second].first != i)
swap(es[e.second].first, es[e.second].second);
            st.emplace(e);
         }
      }
      ord.pop_back();
      reverse(all(ord));
      return ord;
   }

   vector<vector<int>> enumerate_et() {
      if(directed) {
         for(auto& p : deg)
            if(p != 0) return {};
      } else {
         for(auto& p : deg) {
            if(p & 1) return {};
         }
      }
      usedE.assign(M, 0);
      vector<vector<int>> ret;
      for(int i = 0; i < size(G); i++) {
         if(G[i].empty() || usedV[i]) continue;
         ret.emplace_back(go(i));
      }
      return ret;
   }  // a9700f
```

```
   vector<vector<int>> enumerate_semi_et() {
      dsu d(size(G));
      for(auto& p : es) d.merge(p.first, p.second);
      vector<vector<int>> group(size(G));
      for(int i = 0; i < size(G); i++)
group[d.leader(i)].emplace_back(i);
      vector<vector<int>> ret;
      usedE.assign(M, 0);
      for(auto& vs : group) {
         if(vs.empty()) continue;
         int latte = -1, malta = -1;
         if(directed) {
            for(auto& p : vs) {
               if(abs(deg[p]) > 1) return {};
               else if(deg[p] == 1) {
                  if(latte >= 0) return {};
                  latte = p;
               }
            }
         } else {
            for(auto& p : vs) {
               if(deg[p] & 1) {
                  if(latte == -1) latte = p;
                  else if(malta == -1) malta = p;
                  else return {};
               }
            }
         }
         ret.emplace_back(go(latte == -1 ? vs.front() : latte));
         if(ret.back().empty()) ret.pop_back();
      }
      return ret;
   }  // 97a2af

   pair<int, int> get_edge(int i) { return es[i]; }  // c83977
};
```

low_link.hpp md5: 862a6c

```
struct LowLink {
   vector<vector<int>> g;
   vector<int> ord, low;
   vector<int> articulation;
   vector<bool> visited;
   vector<pair<int, int>> bridge;

   void dfs(int cur, int pre, int& k) {
      visited[cur] = true;
      ord[cur] = low[cur] = k++;
      bool isArticulation = false, beet = false;
      int cnt = 0;
      for(auto to : g[cur]) {
         if(to == pre && !exchange(beet, true)) continue;
         if(!visited[to]) {
            cnt++;
            dfs(to, cur, k);
            chmin(low[cur], low[to]);
            isArticulation |= pre != -1 && low[to] >= ord[cur];
            if(ord[cur] < low[to]) bridge.emplace_back(min(cur,
to), max(cur, to));
         } else chmin(low[cur], ord[to]);
      }
      isArticulation |= pre == -1 && cnt > 1;
      if(isArticulation) articulation.push_back(cur);
   }

   void build(const vector<vector<int>>& g) {
      int n = g.size();
      this->g = g;
      ord.assign(n, -1);
      low.assign(n, -1);
      visited.assign(n, false);
      int k = 0;
      for(int i = 0; i < n; i++)
         if(!visited[i]) dfs(i, -1, k);
   }

   LowLink(const vector<vector<int>>& g) { build(g); }
};
```

manhattan_mst.hpp md5: 8b7b06

```
// 候補の辺を O(N) 本に減らす。MST時は追加でsort, UF等の必要あり。
template<typename T> vector<tuple<T, int, int>>
manhattan_mst(vector<T> xs, vector<T> ys) {
   assert(xs.size() == ys.size());
   vector<tuple<T, int, int>> ret;
   int n = (int)xs.size();
   vector<int> ord(n);
   iota(ord.begin(), ord.end(), 0);

   for(int s = 0; s < 2; s++) {
      for(int t = 0; t < 2; t++) {
         auto cmp = [&](int i, int j) -> bool { return xs[i] +
ys[i] < xs[j] + ys[j]; };
         sort(ord.begin(), ord.end(), cmp);

         map<T, int> idx;
         for(int i : ord) {
            for(auto it = idx.lower_bound(-ys[i]); it != idx.end();
it = idx.erase(it)) {
               int j = it->second;
               if(xs[i] - xs[j] < ys[i] - ys[j]) break;
               ret.emplace_back(abs(xs[i] - xs[j]) + abs(ys[i] -
ys[j]), i, j);
            }
            idx[-ys[i]] = i;
         }
         swap(xs, ys);
      }
      for(int i = 0; i < n; i++) xs[i] *= -1;
   }
   return ret;
}
```

max_flow.hpp md5: a7f1d5

```
// base: 9927a4
template<class Cap> struct mf_graph {
   public:
   mf_graph() : _n(0) {}
   mf_graph(int n) : _n(n), g(n) {}

   int add_edge(int from, int to, Cap cap) {
      // assert(0 <= from && from < _n);
      // assert(0 <= to && to < _n);
      // assert(0 <= cap);
      int m = size(pos);
      pos.push_back({from, size(g[from])});
      int from_id = size(g[from]);
      int to_id = size(g[to]);
      if(from == to) to_id++;
      g[from].push_back(_edge{to, to_id, cap});
      g[to].push_back(_edge{from, from_id, 0});
      return m;
   }

   Cap flow(int s, int t, Cap flow_limit =
numeric_limits<Cap>::max()) {
      // assert(0 <= s && s < _n);
      // assert(0 <= t && t < _n);
      // assert(s != t);

      vector<int> level(_n), iter(_n);
      queue<int> que;
      auto bfs = [&]() {
         fill(all(level), -1);
         level[s] = 0;
         while(!que.empty()) que.pop();
         que.push(s);
         while(!que.empty()) {
            int v = que.front();
            que.pop();
            for(auto e : g[v]) {
               if(e.cap == 0 || level[e.to] >= 0) continue;
               level[e.to] = level[v] + 1;
               if(e.to == t) return;
               que.push(e.to);
            }
         }
      };
      auto dfs = [&](auto self, int v, Cap up) {
         if(v == s) return up;
         Cap res = 0;
         int level_v = level[v];
         for(int& i = iter[v]; i < size(g[v]); i++) {
            _edge& e = g[v][i];
            if(level_v <= level[e.to] || g[e.to][e.rev].cap == 0)
continue;
            Cap d = self(self, e.to, min(up - res, g[e.to]
[e.rev].cap));
            if(d <= 0) continue;
            g[v][i].cap += d;
            g[e.to][e.rev].cap -= d;
            res += d;
            if(res == up) break;
```

```
        }
        return res;
    };

    Cap flow = 0;
    while(flow < flow_limit) {
        bfs();
        if(level[t] == -1) break;
        fill(all(iter), 0);
        while(flow < flow_limit) {
            Cap f = dfs(dfs, t, flow_limit - flow);
            if(!f) break;
            flow += f;
        }
    }
    return flow;
}

vector<bool> min_cut(int s) {
    vector<bool> visited(_n);
    queue<int> que;
    que.push(s);
    visited[s] = true;
    while(!que.empty()) {
        int v = que.front();
        que.pop();
        for(auto e : g[v]) {
            if(e.cap && !visited[e.to]) {
                visited[e.to] = true;
                que.push(e.to);
            }
        }
    }
    return visited;
}  // 8735cf

struct edge {
    int from, to;
    Cap cap, flow;
};  // 9fe107

edge get_edge(int i) {
    int m = size(pos);
    // assert(0 <= i && i < m);
    auto _e = g[pos[i].first][pos[i].second];
    auto _re = g[_e.to][_e.rev];
    return edge{pos[i].first, _e.to, _e.cap + _re.cap, _re.cap};
}  // ad4299

vector<edge> edges() {
    int m = size(pos);
    vector<edge> result;
    for(int i = 0; i < m; i++) result.push_back(get_edge(i));
    return result;
}  // 5948b8

void change_edge(int i, Cap new_cap, Cap new_flow) {
    int m = size(pos);
    // assert(0 <= i && i < m);
    // assert(0 <= new_flow && new_flow <= new_cap);
    auto& _e = g[pos[i].first][pos[i].second];
    auto& _re = g[_e.to][_e.rev];
    _e.cap = new_cap - new_flow;
    _re.cap = new_flow;
}  // 558c35

private:
int _n;
struct _edge {
    int to, rev;
    Cap cap;
};
vector<pair<int, int>> pos;
vector<vector<_edge>> g;
};
```

min_cost_flow.hpp md5: 17d51b

```
// base: 4e9f1c
template<class Cap, class Cost> struct mcf_graph {
public:
mcf_graph() {}
mcf_graph(int n) : _n(n), g(n) {}

int add_edge(int from, int to, Cap cap, Cost cost) {
    // assert(0 <= from && from < _n);
    // assert(0 <= to && to < _n);
    int m = size(pos);
    pos.push_back({from, size(g[from])});
    int from_id = size(g[from]);
    int to_id = size(g[to]);
    if(from == to) to_id++;
    g[from].push_back(_edge{to, to_id, cap, cost});
    g[to].push_back(_edge{from, from_id, 0, -cost});
    return m;
}

pair<Cap, Cost> flow(int s, int t, Cap flow_limit =
numeric_limits<Cap>::max()) {
    return slope(s, t, flow_limit).back();
}
vector<pair<Cap, Cost>> slope(int s, int t, Cap flow_limit =
numeric_limits<Cap>::max()) {
    // assert(0 <= s && s < _n);
    // assert(0 <= t && t < _n);
    // assert(s != t);
    vector<Cost> dual(_n, 0), dist(_n);
    vector<int> pv(_n), pe(_n);
    vector<bool> vis(_n);
    auto dual_ref = [&]() {
        fill(all(dist), numeric_limits<Cost>::max());
        fill(all(pv), -1);
        fill(all(pe), -1);
        fill(all(vis), false);
        struct Q {
            Cost key;
            int to;
            bool operator<(const Q& r) const { return key > r.key;
}
        };
        priority_queue<Q> que;
        dist[s] = 0;
        que.push(Q{0, s});
        while(!que.empty()) {
            int v = que.top().to;
            que.pop();
            if(vis[v]) continue;
            vis[v] = true;
            if(v == t) break;
            for(int i = 0; i < size(g[v]); i++) {
                auto e = g[v][i];
                if(vis[e.to] || !e.cap) continue;
                Cost cost = e.cost - dual[e.to] + dual[v];
                if(chmin(dist[e.to], dist[v] + cost)) {
                    pv[e.to] = v;
                    pe[e.to] = i;
                    que.push(Q{dist[e.to], e.to});
                }
            }
        }
        if(!vis[t]) return false;
        for(int v = 0; v < _n; v++)
            if(vis[v]) dual[v] -= dist[t] - dist[v];
        return true;
    };
    Cap flow = 0;
    Cap cost = 0, prev_cost_per_flow = -1;
    vector<pair<Cap, Cost>> result;
    result.push_back({flow, cost});
    while(flow < flow_limit) {
        if(!dual_ref()) break;
        Cap c = flow_limit - flow;
        for(int v = t; v != s; v = pv[v]) { c = min(c, g[pv[v]]
[pe[v]].cap); }
        for(int v = t; v != s; v = pv[v]) {
            auto& e = g[pv[v]][pe[v]];
            e.cap -= c;
            g[v][e.rev].cap += c;
        }
        Cost d = -dual[s];
        flow += c;
        cost += c * d;
        if(prev_cost_per_flow == d) { result.pop_back(); }
        result.push_back({flow, cost});
        prev_cost_per_flow = d;
    }
    return result;
}

struct edge {
    int from, to;
    Cap cap, flow;
};  // 9fe107

edge get_edge(int i) {
    int m = size(pos);
```

```
      // assert(0 <= i && i < m);
      auto _e = g[pos[i].first][pos[i].second];
      auto _re = g[_e.to][_e.rev];
      return edge({pos[i].first, _e.to, _e.cap + _re.cap,
_re.cap});
   }  // d7bd7e

   vector<edge> edges() {
      int m = size(pos);
      vector<edge> result;
      for(int i = 0; i < m; i++) result.push_back(get_edge(i));
      return result;
   }  // 5948b8

   void change_edge(int i, Cap new_cap, Cap new_flow) {
      int m = size(pos);
      // assert(0 <= i && i < m);
      // assert(0 <= new_flow && new_flow <= new_cap);

      auto& _e = g[pos[i].first][pos[i].second];
      auto& _re = g[_e.to][_e.rev];
      _e.cap = new_cap - new_flow;
      _re.cap = new_flow;
   }  // 558c35

   private:
   int _n;
   struct _edge {
      int to, rev;
      Cap cap;
      Cost cost;
   };

   vector<pair<int, int>> pos;
   vector<vector<_edge>> g;
};
```

scc.hpp md5: 9f5fd6

```
// base: 3085f6
struct scc_graph {
   public:
   explicit scc_graph(int _n = 0) : n(_n), G(_n), rG(_n), comp(_n,
-1), visited(_n, 0) {}

   void add_edge(int from, int to) {
      // assert(0 <= from && from < n);
      // assert(0 <= to && to < n);
      G[from].push_back(to);
      rG[to].push_back(from);
   }

   vector<vector<int>> scc() {
      fill(all(visited), 0);
      fill(all(comp), -1);
      order.clear();
      for(int i = 0; i < n; i++)
         if(!visited[i]) dfs(i);
      comp_size = 0;
      for(int i = size(order) - 1; i >= 0; i--) {
         if(comp[order[i]] < 0) rdfs(order[i], comp_size++);
      }
      vector<vector<int>> v(comp_size);
      for(int i = 0; i < n; i++) v[comp[i]].push_back(i);
      return v;
   }

   vector<int> get_comp() { return comp; }  // bdafc0

   vector<vector<int>> dag() {
      vector<vector<int>> res(comp_size);
      for(int i = 0; i < n; i++)
         for(auto j : G[i]) {
            if(comp[i] != comp[j]) res[comp[i]].push_back(comp[j]);
         }
      for(int i = 0; i < comp_size; i++) {
         sort(all(res[i]));
         res[i].erase(unique(all(res[i])), res[i].end());
      }
      return res;
   }  // 312650

   private:
   vector<vector<int>> G, rG;
   vector<int> order, comp;
   vector<bool> visited;
   int n, comp_size;

   void dfs(int v) {
      visited[v] = true;
      for(auto to : G[v])
         if(!visited[to]) dfs(to);
      order.push_back(v);
   }

   void rdfs(int v, int k) {
      comp[v] = k;
      for(auto to : rG[v]) {
         if(comp[to] < 0) rdfs(to, k);
      }
   }
};
```

two_sat.hpp md5: 681721

```
struct two_sat {
   public:
   two_sat() : _n(0), scc(0) {}
   two_sat(int n) : _n(n), scc(2 * n), _answer(n) {}

   void add_clause(int i, bool f, int j, bool g) {
      // assert(0 <= i && i < _n);
      // assert(0 <= j && j < _n);
      scc.add_edge(2 * i + (f ? 0 : 1), 2 * j + (g ? 1 : 0));
      scc.add_edge(2 * j + (g ? 0 : 1), 2 * i + (f ? 1 : 0));
   }

   bool satisfiable() {
      scc.scc();
      auto comp = scc.get_comp();
      for(int i = 0; i < _n; i++) {
         if(comp[2 * i] == comp[2 * i + 1]) return false;
         _answer[i] = comp[2 * i] < comp[2 * i + 1];
      }
      return true;
   }

   vector<bool> answer() { return _answer; }

   private:
   int _n;
   vector<bool> _answer;
   scc_graph scc;
};
```

graph/tree

cartesian_tree.hpp md5: ac77a5

```
template<class T> struct cartesian_tree {
   int root;
   vector<int> par, left, right;

   cartesian_tree(const vector<T>& v) : root(0), par(size(v), -1),
left(size(v), -1), right(size(v), -1) {
      stack<int> st;
      int N = size(v);
      for(int i = 0; i < N; i++) {
         int prev = -1;
         while(!st.empty() && v[st.top()] > v[i]) {
            prev = st.top();
            st.pop();
         }
         if(prev != -1) par[prev] = i;
         if(!st.empty()) par[i] = st.top();
         st.push(i);
      }

      root = -1;
      for(int i = 0; i < N; i++) {
         if(par[i] == -1) root = i;
         else if(par[i] < i) right[par[i]] = i;
         else left[par[i]] = i;
      }
   }
};
```

dominator_tree.hpp md5: 1c5d94

```
struct DominatorTree {
   public:
   DominatorTree(vector<vector<int>>& g_, int root = 0) : g(g_) {
      const int N = (int)g.size();
      rg = vector<vector<int>>(N);
      par.assign(N, 0);
      idom.assign(N, -1);
```

```
      semi.assign(N, -1);
      ord.reserve(N);
      UnionFind uf(semi);

      dfs(root);
      for(int i = 0; i < N; i++) {
         for(auto& to : g[i]) {
            if(~semi[i]) rg[to].emplace_back(i);
         }
      }

      vector<vector<int>> bucket(N);
      vector<int> U(N);
      for(int i = (int)ord.size() - 1; i >= 0; i--) {
         int x = ord[i];
         for(int v : rg[x]) {
            v = uf.eval(v);
            if(semi[x] > semi[v]) semi[x] = semi[v];
         }
         bucket[ord[semi[x]]].emplace_back(x);
         for(int v : bucket[par[x]]) U[v] = uf.eval(v);
         bucket[par[x]].clear();
         uf.link(par[x], x);
      }
      for(int i = 1; i < (int)ord.size(); i++) {
         int x = ord[i], u = U[x];
         idom[x] = semi[x] == semi[u] ? semi[x] : idom[u];
      }
      for(int i = 1; i < (int)ord.size(); i++) {
         int x = ord[i];
         idom[x] = ord[idom[x]];
      }
      idom[root] = root;
   }

   int operator[](const int& k) const { return idom[k]; }

   private:
   vector<vector<int>> g, rg;

   struct UnionFind {
      const vector<int>& semi;
      vector<int> par, m;

      explicit UnionFind(const vector<int>& semi) : semi(semi),
par(semi.size()), m(semi.size()) {
         iota(begin(par), end(par), 0);
         iota(begin(m), end(m), 0);
      }

      int find(int v) {
         if(par[v] == v) return v;
         int r = find(par[v]);
         if(semi[m[v]] > semi[m[par[v]]]) m[v] = m[par[v]];
         return par[v] = r;
      }

      int eval(int v) {
         find(v);
         return m[v];
      }

      void link(int p, int c) { par[c] = p; }
   };

   vector<int> ord, par;
   vector<int> idom, semi;

   void dfs(int idx) {
      semi[idx] = (int)ord.size();
      ord.emplace_back(idx);
      for(auto& to : g[idx]) {
         if(~semi[to]) continue;
         dfs(to);
         par[to] = idx;
      }
   }
};
```

hld.hpp md5: 10247f

```
class HLDcomposition {
   private:
   int V;
   vector<vector<int>> G;
   vector<int> stsize, parent, pathtop, in, out;
   int root;
   void build_stsize(int u, int p) {
      stsize[u] = 1, parent[u] = p;
      for(auto&& v : G[u]) {
         if(v == p) {
            if(v == G[u].back()) break;
            else swap(v, G[u].back());
         }
         build_stsize(v, u);
         stsize[u] += stsize[v];
         if(stsize[v] > stsize[G[u][0]]) swap(v, G[u][0]);
      }
   }

   void build_path(int u, int p, int& tm) {
      in[u] = tm++;
      for(auto v : G[u]) {
         if(v == p) continue;
         pathtop[v] = (v == G[u][0] ? pathtop[u] : v);
         build_path(v, u, tm);
      }
      out[u] = tm;
   }

   public:
   void add_edge(int u, int v) {
      G[u].push_back(v);
      G[v].push_back(u);
   }

   void build(int _root = 0) {
      root = _root;
      int tm = 0;
      build_stsize(root, -1);
      pathtop[root] = root;
      build_path(root, -1, tm);
   }

   inline int index(int a) { return in[a]; }

   int lca(int a, int b) {
      int pa = pathtop[a], pb = pathtop[b];
      while(pa != pb) {
         if(in[pa] > in[pb]) {
            a = parent[pa], pa = pathtop[a];
         } else {
            b = parent[pb], pb = pathtop[b];
         }
      }
      if(in[a] > in[b]) swap(a, b);
      return a;
   }

   pair<int, int> subtree_query(int a) { return {in[a], out[a]}; }

   vector<tuple<int, int, bool>> path_query(int from, int to) {
      int pf = pathtop[from], pt = pathtop[to];
      using T = tuple<int, int, bool>;
      deque<T> front, back;
      while(pf != pt) {
         if(in[pf] > in[pt]) {
            front.push_back({in[pf], in[from] + 1, true});
            from = parent[pf], pf = pathtop[from];
         } else {
            back.push_front({in[pt], in[to] + 1, false});
            to = parent[pt], pt = pathtop[to];
         }
      }
      if(in[from] > in[to]) front.push_back({in[to], in[from] + 1,
true});
      else front.push_back({in[from], in[to] + 1, false});
      vector<T> ret;
      while(!front.empty()) {
         ret.push_back(front.front());
         front.pop_front();
      }
      while(!back.empty()) {
         ret.push_back(back.front());
         back.pop_front();
      }
      return ret;
   }

   HLDcomposition(int node_size)
       : V(node_size), G(V), stsize(V, 0), parent(V, -1),
pathtop(V, -1), in(V, -1), out(V, -1) {}
};
```

flow

燃やす埋める.md

変形前の制約	変形後の制約
$x, y, \ldots$ 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: 298f79

```
// kmp[i] := max{ l ≤ i | s[:l] == s[(i+1)-l:i+1] }
// abacaba -> 0010123
auto KMP(string s) {
   vector<ll> p(size(s));
   for(int i = 1; i < size(s); i++) {
      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: 20c548

```
// 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 = size(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: 7403a8

```
// using u64 = uint64_t;
const u64 mod = INF;
u64 add(u64 a, u64 b) {
   a += b;
   if(a >= mod) a -= mod;
   return a;
}
u64 mul(u64 a, u64 b) {
   auto c = (__uint128_t)a * b;
   return add(c >> 61, c & mod);
}
random_device rnd;
const u64 r = ((u64)rnd() << 32 | rnd()) % mod;
struct RH {
   ll n;
   vector<u64> hs, pw;
   RH(string s) : n(size(s)), hs(n + 1), pw(n + 1, 1) {
      for(int i = 0; i < n; i++) {
         pw[i + 1] = mul(pw[i], r);
         hs[i + 1] = add(mul(hs[i], r), s[i]);
      }
   }
   u64 get(ll l, ll r) const { return add(hs[r], mod - mul(hs[l],
pw[r - l])); }
};
```

SuffixArray.hpp

md5: cbf1dc

```
// 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 = size(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);
      for(int i = 0; i < n; i++)
         if(sa[i] >= j) y[p++] = sa[i] - j;
      fill(all(ws), 0);
      for(int i = 0; i < n; i++) ws[x[i]]++;
      for(int i = 1; i < lim; i++) 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;
      for(int i = 1; i < n; i++) {
         ll a = sa[i - 1], b = sa[i];
         x[b] = (y[a] == y[b] && y[a + j] == y[b + j]) ? p - 1 :
p++;
      }
   }
   for(int i = 1; i < n; i++) rk[sa[i]] = i;
   for(int 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: 6388f3

```
// Z[i] := LCP(s, s[i:])
// abacaba -> 7010301
auto Z(string s) {
   ll n = size(s), l = -1, r = -1;
   vector<ll> z(n, n);
   for(int i = 1; i < n; i++) {
      ll& x = z[i] = i < r ? min(r - i, z[i - l]) : 0;
      while(i + x < n && s[i + x] == s[x]) x++;
      if(i + x > r) l = i, r = i + x;
   }
   return z;
}
```

aho_corasick.hpp

md5: f30f1f

```
// base: 822702
template<int char_size, int margin> struct AhoCorasick :
Trie<char_size + 1, margin> {
   using Trie<char_size + 1, margin>::Trie;

   const int FAIL = char_size;
   vector<int> correct;

   void build(bool heavy = true) {
      correct.resize(this->nodes.size());
      for(int i = 0; i < size(this->nodes); ++i) { correct[i] =
size(this->nodes[i].accept); }
      queue<int> que;
      for(int i = 0; i < char_size; ++i) {
         if(~this->nodes[0].nxt[i]) {
            this->nodes[this->nodes[0].nxt[i]].nxt[FAIL] = 0;
            que.emplace(this->nodes[0].nxt[i]);
         } else {
            this->nodes[0].nxt[i] = 0;
         }
      }
      while(!que.empty()) {
         auto& now = this->nodes[que.front()];
         int fail = now.nxt[FAIL];
         correct[que.front()] += correct[fail];
         que.pop();
         for(int i = 0; i < char_size; i++) {
            if(~now.nxt[i]) {
               this->nodes[now.nxt[i]].nxt[FAIL] = this-
>nodes[fail].nxt[i];
               if(heavy) {
                  auto& u = this->nodes[now.nxt[i]].accept;
                  auto& v = this->nodes[this-
>nodes[fail].nxt[i]].accept;
                  vector<int> accept;
                  set_union(all(u), all(v), back_inserter(accept));
                  u = accept;
               }
               que.emplace(now.nxt[i]);
            } else {
               now.nxt[i] = this->nodes[fail].nxt[i];
            }
```

```
        }
      }
   }

   vector<int> match(const char& c, int now = 0) {
      vector<int> res;
      now = this->nodes[now].nxt[c - margin];
      for(auto& v : this->nodes[now].accept) res.push_back(v);
      return res;
   }  // 68ef6b

   unordered_map<int, int> match(const string& str, int now = 0) {
      unordered_map<int, int> res, visit_cnt;
      for(auto& c : str) {
         now = this->nodes[now].nxt[c - margin];
         visit_cnt[now]++;
      }
      for(auto& [now, cnt] : visit_cnt) {
         for(auto& v : this->nodes[now].accept) res[v] += cnt;
      }
      return res;
   }  // 36fe6c

   pair<ll, int> move(const char& c, int now = 0) {
      now = this->nodes[now].nxt[c - margin];
      return {correct[now], now};
   }  // 43ccad

   pair<ll, int> move(const string& str, int now = 0) {
      ll res = 0;
      for(auto& c : str) {
         auto [cnt, nxt] = move(c, now);
         res += cnt;
         now = nxt;
      }
      return {res, now};
   }  // b1949a
};
```

sa_is.hpp md5: 162db6

```
vector<int> sa_is(const vector<int>& s, int upper) {
   int n = s.size();
   if(n == 0) return {};
   if(n == 1) return {0};
   if(n == 2) {
      if(s[0] < s[1]) {
         return {0, 1};
      } else {
         return {1, 0};
      }
   }
   vector<int> sa(n);
   vector<bool> ls(n);
   for(int i = n - 2; i >= 0; i--) { ls[i] = (s[i] == s[i + 1]) ?
ls[i + 1] : s[i] < s[i + 1]; }
   vector<int> sum_l(upper + 1), sum_s(upper + 1);
   for(int i = 0; i < n; i++) {
      if(!ls[i]) sum_s[s[i]]++;
      else sum_l[s[i] + 1]++;
   }
   for(int i = 0; i <= upper; i++) {
      sum_s[i] += sum_l[i];
      if(i < upper) sum_l[i + 1] += sum_s[i];
   }

   auto induce = [&](const vector<int>& lms) {
      fill(all(sa), -1);
      vector<int> buf(upper + 1);
      copy(all(sum_s), buf.begin());
      for(auto d : lms) {
         if(d == n) continue;
         sa[buf[s[d]]++] = d;
      }
      copy(all(sum_l), buf.begin());
      sa[buf[s[n - 1]]++] = n - 1;
      for(int i = 0; i < n; i++) {
         int v = sa[i];
         if(v >= 1 && !ls[v - 1]) sa[buf[s[v - 1]]++] = v - 1;
      }
      copy(all(sum_l), buf.begin());
      for(int i = n - 1; i >= 0; i--) {
         int v = sa[i];
         if(v >= 1 && ls[v - 1]) sa[--buf[s[v - 1] + 1]] = v - 1;
      }
   };

   vector<int> lms_map(n + 1, -1);
   int m = 0;
   for(int i = 1; i < n; i++) {
      if(!ls[i - 1] && ls[i]) lms_map[i] = m++;
   }
   vector<int> lms;
   lms.reserve(m);
   for(int i = 1; i < n; i++) {
      if(!ls[i - 1] && ls[i]) lms.push_back(i);
   }
   induce(lms);

   if(m) {
      vector<int> sorted_lms;
      sorted_lms.reserve(m);
      for(int v : sa) {
         if(lms_map[v] != -1) sorted_lms.push_back(v);
      }
      vector<int> rec_s(m);
      int rec_upper = 0;
      rec_s[lms_map[sorted_lms[0]]] = 0;
      for(int i = 1; i < m; i++) {
         int l = sorted_lms[i - 1], r = sorted_lms[i];
         int end_l = (lms_map[l] + 1 < m) ? lms[lms_map[l] + 1] :
n;
         int end_r = (lms_map[r] + 1 < m) ? lms[lms_map[r] + 1] :
n;
         bool same = true;
         if(end_l - l != end_r - r) same = false;
         else {
            while(l < end_l) {
               if(s[l] != s[r]) break;
               l++;
               r++;
            }
            if(l == n || s[l] != s[r]) same = false;
         }
         if(!same) rec_upper++;
         rec_s[lms_map[sorted_lms[i]]] = rec_upper;
      }

      auto rec_sa = sa_is(rec_s, rec_upper);

      for(int i = 0; i < m; i++) { sorted_lms[i] = lms[rec_sa[i]];
}
      induce(sorted_lms);
   }
   return sa;
}
```

trie.hpp md5: 09415e

```
template<int char_size> struct TrieNode {
   int nxt[char_size];
   int exist;
   vector<int> accept;

   TrieNode() : exist(0) { memset(nxt, -1, sizeof(nxt)); }
};

template<int char_size, int margin> struct Trie {
   using Node = TrieNode<char_size>;

   vector<Node> nodes;
   int root;
   Trie() : root(0) { nodes.push_back(Node()); }

   void update_direct(int node, int id) {
nodes[node].accept.push_back(id); }

   void update_child(int node, int child, int id) {
++nodes[node].exist; }

   void add(const string& str, int str_index, int node_index, int
id) {
      if(str_index == size(str)) {
         update_direct(node_index, id);
      } else {
         const int c = str[str_index] - margin;
         if(nodes[node_index].nxt[c] == -1) {
            nodes[node_index].nxt[c] = size(nodes);
            nodes.push_back(Node());
         }
         add(str, str_index + 1, nodes[node_index].nxt[c], id);
         update_child(node_index, nodes[node_index].nxt[c], id);
      }
   }

   void add(const string& str, int id = -1) {
```

```
      if(id == -1) id = nodes[0].exist;
      add(str, 0, 0, id);
   }

   void query(const string& str, const function<void(int)>& f, int
str_index, int node_index) {
      for(auto& idx : nodes[node_index].accept) f(idx);
      if(str_index == size(str)) {
         return;
      } else {
         const int c = str[str_index] - margin;
         if(nodes[node_index].nxt[c] == -1) return;
         query(str, f, str_index + 1, nodes[node_index].nxt[c]);
      }
   }

   void query(const string& str, const function<void(int)>& f) {
query(str, f, 0, 0); }

   int count() const { return nodes[0].exist; }
};
```

algorithm

3d_mo.hpp

md5: 85daf7

```
struct Mo_3D {
   int width;
   vector<int> left, right, index, order;
   vector<bool> v;
   function<void(int)> add, del;

   Mo_3D(int N, int Q) : order(Q), v(N) { width = max<int>(1,
pow(N, 2.0 / 3.0)); }
   void insert(int idx, int l, int r) {
      index.push_back(idx);
      left.emplace_back(l);
      right.emplace_back(r);
   }

   void run(const auto& add, const auto& del, const auto& rem,
const auto& add_query, const auto& del_query) {
      this->add = add;
      this->del = del;
      order.resize(size(left));
      iota(all(order), 0);
      sort(all(order), [&](int a, int b) -> bool {
         if(left[a] / width != left[b] / width) return left[a] <
left[b];
         if(right[a] / width != right[b] / width) return ((right[a]
< right[b]) ^ (left[a] / width % 2));
         return bool((index[a] < index[b]) ^ (right[a] / width %
2));
      });
      int time = 0, nl = 0, nr = 0;
      for(auto idx : order) {
         while(time < index[idx]) add_query(time++, this);
         while(time > index[idx]) del_query(--time, this);
         while(nl > left[idx]) distribute(--nl);
         while(nr < right[idx]) distribute(nr++);
         while(nl < left[idx]) distribute(nl++);
         while(nr > right[idx]) distribute(--nr);
         rem(index[idx]);
      }
   }

   void distribute(int idx) {
      if(v[idx]) del(idx);
      else add(idx);
      v[idx] = !v[idx];
   }
};
```

larsch.hpp

md5: ba650a

```
template<typename T> vector<T> monge_shortest_path(int N, const
function<T(int, int)>& f) {
   T INF = (T{1} << (sizeof(T) * 8 - 2)) - 1;
   vector<T> dp(N + 1, INF);
   vector<int> x(N + 1, 0);
   auto check = [&](int from, int to) {
      if(from >= to) return;
      T cost = f(from, to);
      if(dp[from] + cost < dp[to]) dp[to] = dp[from] + cost, x[to]
= from;
   };
   auto dfs = [&](auto rc, int l, int r) -> void {
      if(l + 1 >= r) return;
      int m = (l + r) / 2;
      for(int i = x[l]; i <= x[r]; i++) check(i, m);
      rc(rc, l, m);
      for(int i = l + 1; i <= m; i++) check(i, r);
      rc(rc, m, r);
   };
   dp[0] = 0, check(0, N), dfs(dfs, 0, N);
   return dp;
}

// [min, max] は閉区間を入力する
template<typename T, bool get_min = true> pair<ll, T>
golden_section_search(const function<T(ll)>& f, ll min, ll max) {
   assert(min <= max);
   ll a = min - 1, x, b;
   {
      ll s = 1, t = 2;
      while(t < max - min + 2) swap(s += t, t);
      x = a + t - s, b = a + t;
   }
   T fx = f(x), fy;
   while(a + b != 2 * x) {
      ll y = a + b - x;
      if(max < y || (fy = f(y), get_min ? fx < fy : fx > fy)) {
         b = a;
         a = y;
      } else {
         a = x;
         x = y;
         fx = fy;
      }
   }
   return {x, fx};
}

// upper : max abs(辺数を 1 増減させたときのコストの変化)
ll monge_d_edge_shortest_path(int N, int D, ll upper, const
function<ll(int, int)>& f) {
   using T = __int128_t;
   upper = abs(upper);
   auto dp = [&](ll x) -> T {
      auto g = [&](int from, int to) -> T { return f(from, to) + x;
};
      T cost = monge_shortest_path<T>(N, g)[N];
      return cost - T{1} * D * x;
   };
   auto [_, res] = golden_section_search<T, false>(dp, -upper,
upper);
   return res;
}

vector<ll> enumerate_monge_d_edge_shortest_path(int N,
                                                const
function<ll(int, int)>& f,
                                                ll unreached = (1LL
<< 62) - 1) {
   using T = __int128_t;
   T INF = (T{1} << (sizeof(T) * 8 - 2)) - 1;
   vector<ll> ans(N + 1, unreached);
   vector<T> dp(N + 1, INF);
   dp[0] = 0;
   for(int d = 1; d <= N; d++) {
      vector<int> midx =
          monotone_minima<T>(N + 1, N + 1, [&](int j, int i) -> T {
return i < j ? dp[i] + f(i, j) : INF; });
      for(int i = N; i >= d; i--) dp[i] = dp[midx[i]] + f(midx[i],
i);
      ans[d] = dp[N];
   }
   return ans;
}
```

min_plus_concave.hpp

md5: c4cd24

```
// b is concave
template<typename T> vector<T> minplus_conv_concave(vector<T>& a,
vector<T>& b) {
   int n = a.size(), m = b.size();
   if(min(n, m) == 0) return vector<T>();
   int h = n + m - 1, w = n;
   vector<int> ymin(h, 0), ymax(h, w - 1), xmin(w), xmax(w);
   for(int x = m; x < h; x++) ymin[x] = x - m + 1;
   for(int x = 0; x <= h - m; x++) ymax[x] = x;
   iota(all(xmin), 0);
   iota(all(xmax), m - 1);
   vector<T> c(h, INF);  // if long long
   auto rec = [&](auto&& rec, int x1, int x2, int y1, int y2) {
      if(ymax[x1] >= y2 and y1 >= ymin[x2]) {
         auto A = [&](int i, int j) { return a[y2 - j] + b[x1 + i -
```

```
y2 + j]; };
        auto jmin = monotone_minima<T>(x2 - x1 + 1, y2 - y1 + 1,
A);
        for(int i = x1; i <= x2; i++) chmin(c[i], A(i - x1, jmin[i
- x1]));
        return;
      }
      if((ll)(x2 - x1) * (y2 - y1) < 1000) {
         for(int x = x1; x <= x2; x++)
            for(int y = max(ymin[x], y1); y <= min(ymax[x], y2);
y++) chmin(c[x], a[y] + b[x - y]);
         return;
      }
      if(x2 - x1 > y2 - y1) {
         int xm = (x1 + x2) / 2;
         int ny2 = min(ymax[xm], y2), ny1 = max(ymin[xm], y1);
         if(y1 <= ny2) rec(rec, x1, xm, y1, ny2);
         if(ny1 <= y2) rec(rec, xm + 1, x2, ny1, y2);
      } else {
         int ym = (y1 + y2) / 2;
         int nx2 = min(xmax[ym], x2), nx1 = max(xmin[ym], x1);
         if(x1 <= nx2) rec(rec, x1, nx2, y1, ym);
         if(nx1 <= x2) rec(rec, nx1, x2, ym + 1, y2);
      }
   };
   rec(rec, 0, h - 1, 0, w - 1);
   return c;
}
```

min_plus_conv.hpp md5: f2bb6c

```
template<typename T, typename F> vector<int> 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);
      for(int i = l + 1; i <= r; i++)
         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);
   vector<int> ret;
   for(auto [idx, val] : dp) ret.push_back(idx);
   return ret;
}

template<typename T>  // B is convex. if both A and B are convex,
do greedy.
vector<T> minplus_conv(vector<T>& A, vector<T>& B) {
   int n = A.size(), m = B.size();
   if(n == 0 && m == 0) return {};
   vector<T> C(n + m - 1);

   const ll inf = INF;
   auto select = [&](int i, int j) -> T {
      if(i < j) return inf;
      if(i - j >= m) return inf;
      return A[j] + B[i - j];
   };
   vector<int> J = monotone_minima<T>(n + m - 1, n, select);
   for(int i = 0; i < n + m - 1; i++) {
      T x = A[J[i]], y = B[i - J[i]];
      if(x < inf && y < inf) C[i] = x + y;
   }
   return C;
}
```

mo.hpp md5: 62406b

```
struct Mo {
   int width;
   vector<int> left, right, order;
   Mo(int N, int Q) : order(Q) {
      width = max<int>(1, 1.0 * N / max<double>(1.0, sqrt(Q * 2.0 /
3.0)));
      iota(all(order), 0);
   }

   void insert(int l, int r) {
      left.emplace_back(l);
      right.emplace_back(r);
   }

   void run(const auto& add_left,
            const auto& add_right,
            const auto& delete_left,
            const auto& delete_right,
            const auto& rem) {
      sort(begin(order), end(order), [&](int a, int b) {
         int ablock = left[a] / width, bblock = left[b] / width;
         if(ablock != bblock) return ablock < bblock;
         return (ablock & 1) ? right[a] > right[b] : right[a] <
right[b];
      });
      int nl = 0, nr = 0;
      for(auto idx : order) {
         while(nl > left[idx]) add_left(--nl);
         while(nr < right[idx]) add_right(nr++);
         while(nl < left[idx]) delete_left(nl++);
         while(nr > right[idx]) delete_right(--nr);
         rem(idx);
      }
   }
};
```

rollback_mo.hpp md5: 8d34d2

```
struct Mo_rollback {
   int width;
   vector<int> left, right, order;
   Mo_rollback(int N, int Q) : order(Q) {
      width = sqrt(N);
      iota(all(order), 0);
   }

   void insert(int l, int r) {
      left.emplace_back(l);
      right.emplace_back(r);
   }

   void run(const auto& add_left,
            const auto& add_right,
            const auto& rem,
            const auto& reset,
            const auto& snapshot,
            const auto& rollback) {
      sort(begin(order), end(order), [&](int a, int b) {
         int ablock = left[a] / width, bblock = left[b] / width;
         if(ablock != bblock) return ablock < bblock;
         return right[a] < right[b];
      });
      reset();
      snapshot();
      for(auto idx : order) {
         if(right[idx] - left[idx] < width) {
            for(int i = left[idx]; i < right[idx]; i++)
add_right(i);
            rem(idx);
            rollback();
         }
      }
      int nr = 0, last_block = -1;
      for(auto idx : order) {
         if(right[idx] - left[idx] < width) continue;
         int block = left[idx] / width;
         if(block != last_block) {
            reset();
            nr = (block + 1) * width;
            last_block = block;
         }
         while(nr < right[idx]) add_right(nr++);
         snapshot();
         for(int j = (block + 1) * width - 1; j >= left[idx]; j--)
add_left(j);
         rem(idx);
         rollback();
      }
   }
};
```

geometry

base.hpp md5: 9ca2e3

```
using Point = complex<double>;
using Line = vector<Point>;

#define X real()
#define Y imag()
const double EPS = 1e-10;

inline double dot(const Point& a, const Point& b) { return a.X *
b.X + a.Y * b.Y; }
inline double cross(const Point& a, const Point& b) { return a.X *
```

```
b.Y - a.Y * b.X; }
inline double abs(const Point& a) { return sqrt(dot(a, a)); }
```

convex_hull.hpp md5: 17af99

```
vector<Point> convex_hull(vector<Point>& ps, bool collinear =
false) {
   int n = ps.size();
   if(n <= 1) return ps;
   sort(ps.begin(), ps.end(),
        [&EPS](const Point& a, const Point& b) { return abs(a.X -
b.X) > EPS ? a.X < b.X : a.Y < b.Y; });
   vector<Point> hull(2 * n);
   double th = collinear ? -EPS : EPS;
   int k = 0;
   for(int i = 0; i < n; i++) {
      if(k >= 2) {
         while(cross(hull[k - 1] - hull[k - 2], ps[i] - hull[k -
2]) < th) {
            k--;
            if(k < 2) break;
         }
      }
      if(k < 1 || abs(hull[k - 1] - ps[i]) > EPS) {
         hull[k] = ps[i];
         k++;
      }
   }
   int t = k + 1;
   for(int i = n - 2; i >= 0; i--) {
      if(k >= t) {
         while(cross(hull[k - 1] - hull[k - 2], ps[i] - hull[k -
2]) < th) {
            k--;
            if(k < t) break;
         }
      }
      if(k < 1 || abs(hull[k - 1] - ps[i]) > EPS) {
         hull[k] = ps[i];
         k++;
      }
   }
   hull.resize(max(1, k - 1));
   return hull;
}
```

etc.hpp md5: 093f7a

```
// 1: a-bから見てa-cが反時計回り
// -1: a-bから見てa-cが時計回り
// 0: a-c-bがこの順で直線
// 2: c-a-bの順で直線
// -2: a-b-cの順で直線

int ccw(const Point& a, const Point& b, const Point& c) {
   if(cross(b - a, c - a) > EPS) return 1;
   if(cross(b - a, c - a) < -EPS) return -1;
   if(dot(b - a, c - a) < -EPS) return 2;
   if(norm(b - a) < norm(c - a) - EPS) return -2;
   return 0;
}  // 6f1927

Point projection(const Point& p, const Line& l) {
   double t = dot(p - l[0], l[1] - l[0]) / norm(l[1] - l[0]);
   return l[0] + t * (l[1] - l[0]);
}  // b9dbd7

Point reflection(const Point& p, const Line& l) { return 2.0 *
projection(p, l) - p; }
// 65ba76

// point and line intersection
bool isinterPL(const Point& p, const Line& l) { return abs(p -
projection(p, l)) < EPS; }
// e9d393

// point and segment intersection
bool isinterPS(const Point& p, const Line& s) { return ccw(s[0],
s[1], p) == 0; }
// 79c17b

// two lines intersection
bool isinterLL(const Line& l, const Line& m) {
   return abs(cross(l[1] - l[0], m[1] - m[0])) > EPS ||
abs(cross(l[1] - l[0], m[0] - l[0])) < EPS;
}  // b58dbd

// two segments intersection
bool isinterSS(const Line& s, const Line& t) {
   if(norm(s[1] - s[0]) < EPS) return isinterPS(s[0], t);
   if(norm(t[1] - t[0]) < EPS) return isinterPS(t[0], s);
   return (ccw(s[0], s[1], t[0]) * ccw(s[0], s[1], t[1]) <= 0) &&
(ccw(t[0], t[1], s[0]) * ccw(t[0], t[1], s[1]) <= 0);
}  // a499ea

double distancePL(const Point& p, const Line& l) { return abs(p -
projection(p, l)); }
// c77772

double distancePS(const Point& p, const Line& s) {
   Point h = projection(p, s);
   if(isinterPS(h, s)) return abs(p - h);
   return min(abs(p - s[0]), abs(p - s[1]));
}  // 3bd780

double distanceLL(const Line& l, const Line& m) {
   if(isinterLL(l, m)) return 0;
   return distancePL(l[0], m);
}  // ab4ace

double distanceSS(const Line& s, const Line& t) {
   if(isinterSS(s, t)) return 0;
   return min(min(distancePS(s[0], t), distancePS(s[1], t)),
min(distancePS(t[0], s), distancePS(t[1], s)));
}  // c284e5

// if(ans){ x = p->X; y = q->Y} else {cout << "not cross"}
optional<Point> crosspoint(const Line& l, const Line& m) {
   double d = cross(m[1] - m[0], l[1] - l[0]);
   if(abs(d) < EPS) return nullopt;
   return l[0] + (l[1] - l[0]) * cross(m[1] - m[0], m[1] - l[0]) /
d;
}  // 687c0c

double area(const vector<Point>& ps) {
   double res = 0;
   for(int i = 0; i < size(ps); i++) res += cross(ps[i], ps[(i + 1)
% size(ps)]);
   return res / 2;
}  // 3b832b

bool is_convex(vector<Point>& ps) {
   int n = (int)ps.size();
   for(int i = 0; i < n; ++i) {
      if(ccw(ps[i], ps[(i + 1) % n], ps[(i + 2) % n]) == -1) return
false;
   }
   return true;
}  // 52fb34

tuple<double, int, int> diameter(const vector<Point> ps) {
   int n = (int)ps.size();
   int si = 0, sj = 0;
   for(int i = 1; i < n; i++) {
      if(ps[i].Y > ps[si].Y) si = i;
      if(ps[i].Y < ps[sj].Y) sj = i;
   }

   double res = 0;
   int i = si, j = sj;
   int ri = i, rj = j;
   do {
      if(chmax(res, abs(ps[i] - ps[j]))) {
         ri = i;
         rj = j;
      }
      if(cross(ps[(i + 1) % n] - ps[i], ps[(j + 1) % n] - ps[j]) <
0) i = (i + 1) % n;
      else j = (j + 1) % n;
   } while(i != si || j != sj);
   return {res, min(ri, rj), max(ri, rj)};
}  // cae9ad

// 2: inside, 1: border, 0: outside
int contains(const vector<Point>& ps, const Point& p) {
   int n = ps.size();
   bool in = false;
   for(int i = 0; i < n; i++) {
      Point a = ps[i] - p, b = ps[(i + 1) % n] - p;
      if(a.Y > b.Y) swap(a, b);
      if(a.Y <= EPS && EPS < b.Y && cross(a, b) < -EPS) in = !in;
      if(abs(cross(a, b)) < EPS && dot(a, b) < EPS) return 1;
   }
   return in ? 2 : 0;
}  // fd7e87
```

```
tuple<double, int, int> closest_pair(vector<Point> ps) {
   const double INF = 1e18;
   int n = (int)ps.size();
   if(n <= 1) return {INF, -1, -1};

   using P = pair<Point, int>;
   vector<P> V(n);
   for(int i = 0; i < n; i++) V[i] = {ps[i], i};
   sort(begin(V), end(V), [](const P& a, const P& b) {
      if(fabs(a.first.X - b.first.X) > EPS) return a.first.X <
b.first.X;
      else if(fabs(a.first.Y - b.first.Y) > EPS) return a.first.Y <
b.first.Y;
      return a.second < b.second;
   });

   auto rec = [&](auto&& self, auto it, int n) -> tuple<double,
int, int> {
      if(n <= 1) return {INF, -1, -1};
      int m = n / 2;
      double x = it[m].first.X;
      auto [d1, a1, b1] = self(self, it, m);
      auto [d2, a2, b2] = self(self, it + m, n - m);
      double d;
      int a, b;
      if(d1 < d2) {
         d = d1;
         a = a1;
         b = b1;
      } else {
         d = d2;
         a = a2;
         b = b2;
      }

      inplace_merge(it, it + m, it + n, [](const P& a, const P& b)
{ return a.first.Y < b.first.Y; });

      vector<P> vec;
      for(int i = 0; i < n; i++) {
         if(fabs(it[i].first.X - x) >= d) continue;
         for(int j = 0; j < size(vec); j++) {
            double dx = fabs(it[i].first.X - vec[size(vec) - j -
1].first.X);
            double dy = fabs(it[i].first.Y - vec[size(vec) - j -
1].first.Y);
            if(dy >= d) break;
            if(chmin(d, sqrt(dx * dx + dy * dy))) {
               a = it[i].second;
               b = vec[size(vec) - j - 1].second;
            }
         }
         vec.emplace_back(it[i]);
      }
      return {d, a, b};
   };
   auto [d, a, b] = rec(rec, V.begin(), n);
   return {d, min(a, b), max(a, b)};
}  // 12a9dc
```

memo

Primes.md

高度合成数

$\leq n$	10^3	10^4	10^6	10^9	10^{12}	10^{16}	10^{18}
x	840	7560	720720	735135500	-	-	-
$d^0(x)$	32	64	240	1344	10752	41472	103680

bigint.md

```
#include <boost/multiprecision/cpp_int.hpp>
namespace mp = boost::multiprecision;
mp::cpp_int x = 1;
```

または

```
using namespace boost::multiprecision;
cpp_int x;
x.assign("123");
```

ext.cpp

md5: 64e006

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/priority_queue.hpp>
#include <ext/rope>

using namespace __gnu_pbds;
using namespace __gnu_cxx;  // for ext/rope
using namespace std;

int main() {
   tree<int, null_type, less<int>, rb_tree_tag,
tree_order_statistics_node_update> tree;
   tree.insert(1);                        // 1
   tree.insert(10);                       // 1 10
   tree.insert(6);                        // 1 6 10
   tree.insert(7);                        // 1 6 7 10
   tree.insert(4);                        // 1 4 6 7 10
   tree.erase(6);                         // 1 4 7 10
   assert(tree.order_of_key(5) == 2);     // 5未満の要素数
   assert(*tree.find_by_order(2) == 7);  // 0-indexedで2番目の要素

   gp_hash_table<int, int> m;
   m[2] = 5;
   m[1] = 10;
   m[3] = 7;

   __gnu_pbds::priority_queue<int, greater<int>,
rc_binomial_heap_tag> pq;
   auto it = pq.push(1);  // 1
   pq.push(10);           // 1 10
   assert(pq.top() == 1);
   pq.modify(it, 20);  // 10 20
   assert(pq.top() == 10);
   pq.erase_if([](int x) { return x <= 10; });  // 20
   assert(pq.top() == 20);
   assert(pq.size() == 1);
   pq.erase(it);  // empty
   assert(pq.empty());

   // access, insert, erase: O(log n)
   rope<int> v;
   v.insert(0, 1);  // 1
   v.insert(0, 2);  // 2 1
   v.insert(1, 3);  // 2 3 1
   v.insert(2, 4);  // 2 3 4 1
   v.erase(1, 2);   // 2 1
   assert(v.size() == 2);
   assert(v[0] == 2);
   assert(v[1] == 1);

   return 0;
}
```

random.hpp

md5: b4f59b

```
struct rng {
   mt19937 mt;
   rng() : mt(58) {}
   int get(int a, int b) {  // [a, b]
      uniform_int_distribution<int> dist(a, b);
      return dist(mt);
   }
};
```

set_compare.md

```
   auto comp = [&](const int a, const int b) {return a > b;};
   set<int, decltype(comp)> st{comp};
```

数式.md

乱順列

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

$$a_n = n! \sum_{k=2}^{n} \frac{(-1)^k}{(k!)}$$

オイラーの五角数定理

$$\prod_{k=1}^{\infty}(1-x^n) = \sum_{n=-\infty}^{\infty}(-1)^n x^{n(3n-1)/2}$$

メビウスの反転公式

$$g(n) = \sum_{d|n} f(d)$$

行列木定理

- n 頂点 m 辺のグラフの(ラベル付き)全域木の個数
- ラプラシアン行列の任意の余因子の行列式の $(-1)^{i+j}$ 倍

LGV 公式

- DAG および始点集合 A と終点集合 B が与えられる。整合的であるとき、これらを始点・終点とするパスの組であって、どの 2 のパスも頂点を共有しないものの個数
- 整合的: 2 つの始点 $a_i < a_j \in A$ と 2 つの終点 $b_i > b_j \in B$ があるとき、 a_i から b_j のパスと a_j から b_i のパスは必ず交わる
- $x_{i,j}$ を a_i から b_j に向かうパスの個数として、$X = (x_{i,j})$ の行列式

LP 双対

強双対性 : $\max\{cx | x \geq 0, Ax \leq b\} = \min\{yb | y \geq 0, yA \geq c\}$

Tutte 行列

完全マッチング存在条件

$$A_{i,j} = \begin{cases} x_{i,j} & \text{if} (i,j) \in E \text{ and } i < j \\ -x_{i,j} & \text{if} (i,j) \in E \text{ and } i > j \\ 0 & \text{otherwise} \end{cases}$$

の行列式が多項式として 0

ラグランジュの反転公式

- F と G が互いに逆関数、つまり $G(F(x)) = F(G(x)) = x$

$$[x^n]G(x) = \frac{1}{n}[x^{n-1}]\left(\frac{x}{F(x)}\right)^n$$

繰り返し文字列

X, Y は空でない有限文字列

- $X^\infty = Y^\infty \iff XY = YX$
- $lcp(X^\infty, Y^\infty) = lcp(XY, YX)$
- $X^\infty < Y \iff XY < Y$

負の二項定理

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

傾き 1 の 2 本の直線の間を通る最短経路

$y = x + s$ と $y = x + t$ の間を通る(直線上の点は通れる)、(a, b) から (c, d) への最短経路の個数は

$$\sum_k \left(\binom{c+d-a-b}{c-a-k(t-s+2)} - \binom{c+d-a-b}{c-b-k(t-s+2)+t+1} \right)$$

フック長の公式

$$d_\lambda = \frac{n!}{\prod h_\lambda(i,j)}$$

other

- 異なる自然数に分割する方法の個数と奇数に分割する方法の個数が等しい

牛ゲー.md

$x_i - x_j \leq M$ のとき

```
G[j].push_back({M, i})
```
