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: 015ba5

```cpp
#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; }

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

data-structure

2dBIT.hpp

md5: a0e12e

```cpp
template<typename T> struct BinaryIndexedTree2D {
   int H, W;
   vector<vector<T>> bit;

   BinaryIndexedTree2D(int H, int W) : H(H), W(W), bit(H + 3,
vector<T>(W + 3, 0)) {}
   void add(int x, int y, T w) {
      if(x < 0 || x >= H || y < 0 || y >= W) return;
      for(int a = (++y, ++x); a <= H; a += a & -a) {
         for(int b = y; b <= W; b += b & -b) { bit[a][b] += w;
}
      }
   }

   void imos(int x1, int y1, int x2, int y2, T w) {
      add(x1, y1, w);
      add(x2, y2, w);
      add(x1, y2, -w);
      add(x2, y1, -w);
   }

   T sum(int x, int y) {
      if(x < 0 || x >= H || y < 0 || y >= W) return 0;
      if(x >= H) x = H - 1;
      if(y >= W) y = W - 1;
      T ret = 0;
      for(int a = (++y, ++x); a > 0; a -= a & -a) {
         for(int b = y; b > 0; b -= b & -b) { ret += bit[a][b];
}
      }
      return ret;
   }

   T sum(int x1, int y1, int x2, int y2) {
      return sum(x2, y2) - sum(x1 - 1, y2) - sum(x2, y1 - 1) +
sum(x1 - 1, y1 - 1);
   }
};
```

BIT.hpp

md5: 4dd117

```cpp
struct BIT {
   vector<ll> a;
   BIT(ll n) : a(n + 1) {}
   void add(ll i, ll x) {  // A[i] += x
      i++;
      while(i < size(a)) {
         a[i] += x;
         i += i & -i;
      }
   }
   ll sum(ll r) {
      ll s = 0;
      while(r) {
         s += a[r];
         r -= r & -r;
      }
      return s;
   }
   ll sum(ll l, ll r) {  // sum of A[l, r)
      return sum(r) - sum(l);
   }
};
```

FastSet.hpp

md5: 5c5532

```cpp
// using u64 = uint64_t;
const u64 B = 64;
struct FastSet {
   u64 n;
   vector<vector<u64>> a;
   FastSet(u64 n_) : n(n_) {
      do a.emplace_back(n_ = (n_ + B - 1) / B);
      while(n_ > 1);
   }
   // bool operator[](ll i) const { return a[0][i / B] >> (i %
B) & 1; }
   void set(ll i) {
      for(auto& v : a) {
         v[i / B] |= 1ULL << (i % B);
         i /= B;
      }
   }
   void reset(ll i) {
      for(auto& v : a) {
         v[i / B] &= ~(1ULL << (i % B));
         if(v[i / B]) break;
         i /= B;
      }
   }
   ll next(ll i) {  // i を超える最小の要素
      for(int h = 0; h < size(a); h++) {
         i++;
         if(i / B >= size(a[h])) break;
         u64 d = a[h][i / B] >> (i % B);
         if(d) {
            i += countr_zero(d);
            while(h--) i = i * B + countr_zero(a[h][i]);
            return i;
         }
         i /= B;
      }
      return n;
   }
   ll prev(ll i) {  // i より小さい最大の要素
      for(int h = 0; h < size(a); h++) {
         i--;
         if(i < 0) break;
         u64 d = a[h][i / B] << (~i % B);
         if(d) {
            i -= countl_zero(d);
            while(h--) i = i * B + __lg(a[h][i]);
            return i;
         }
         i /= B;
      }
      return -1;
   }
};
```

binary_trie.hpp

md5: af8046

```cpp
// 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
};
```

ch_segtree.hpp

md5: 9f5307

```
class ch_segtree {
   private:
   const ll inf = 1e18;
   int n, n0;
   vector<ll> max_v, smax_v, max_c, min_v, smin_v, min_c, sum,
len, ladd, lval;

   void update_node_max(int k, ll x) {
      sum[k] += (x - max_v[k]) * max_c[k];

      if(max_v[k] == min_v[k]) {
         max_v[k] = min_v[k] = x;
      } else if(max_v[k] == smin_v[k]) {
         max_v[k] = smin_v[k] = x;
      } else {
         max_v[k] = x;
      }

      if(lval[k] != inf && x < lval[k]) { lval[k] = x; }
   }
   void update_node_min(int k, ll x) {
      sum[k] += (x - min_v[k]) * min_c[k];

      if(max_v[k] == min_v[k]) {
         max_v[k] = min_v[k] = x;
      } else if(smax_v[k] == min_v[k]) {
         min_v[k] = smax_v[k] = x;
      } else {
         min_v[k] = x;
      }

      if(lval[k] != inf && lval[k] < x) { lval[k] = x; }
   }

   void push(int k) {
      if(n0 - 1 <= k) return;

      if(lval[k] != inf) {
         updateall(2 * k + 1, lval[k]);
         updateall(2 * k + 2, lval[k]);
         lval[k] = inf;
         return;
      }

      if(ladd[k] != 0) {
         addall(2 * k + 1, ladd[k]);
         addall(2 * k + 2, ladd[k]);
         ladd[k] = 0;
      }

      if(max_v[k] < max_v[2 * k + 1]) { update_node_max(2 * k +
1, max_v[k]); }
      if(min_v[2 * k + 1] < min_v[k]) { update_node_min(2 * k +
1, min_v[k]); }

      if(max_v[k] < max_v[2 * k + 2]) { update_node_max(2 * k +
2, max_v[k]); }
      if(min_v[2 * k + 2] < min_v[k]) { update_node_min(2 * k +
2, min_v[k]); }
   }

   void update(int k) {
      sum[k] = sum[2 * k + 1] + sum[2 * k + 2];

      if(max_v[2 * k + 1] < max_v[2 * k + 2]) {
         max_v[k] = max_v[2 * k + 2];
         max_c[k] = max_c[2 * k + 2];
         smax_v[k] = max(max_v[2 * k + 1], smax_v[2 * k + 2]);
      } else if(max_v[2 * k + 1] > max_v[2 * k + 2]) {
         max_v[k] = max_v[2 * k + 1];
         max_c[k] = max_c[2 * k + 1];
         smax_v[k] = max(smax_v[2 * k + 1], max_v[2 * k + 2]);
      } else {
         max_v[k] = max_v[2 * k + 1];
         max_c[k] = max_c[2 * k + 1] + max_c[2 * k + 2];
         smax_v[k] = max(smax_v[2 * k + 1], smax_v[2 * k + 2]);
      }

      if(min_v[2 * k + 1] < min_v[2 * k + 2]) {
         min_v[k] = min_v[2 * k + 1];
         min_c[k] = min_c[2 * k + 1];
         smin_v[k] = min(smin_v[2 * k + 1], min_v[2 * k + 2]);
      } else if(min_v[2 * k + 1] > min_v[2 * k + 2]) {
         min_v[k] = min_v[2 * k + 2];
         min_c[k] = min_c[2 * k + 2];
         smin_v[k] = min(min_v[2 * k + 1], smin_v[2 * k + 2]);
      } else {
         min_v[k] = min_v[2 * k + 1];
         min_c[k] = min_c[2 * k + 1] + min_c[2 * k + 2];
         smin_v[k] = min(smin_v[2 * k + 1], smin_v[2 * k + 2]);
      }
   }

   void _update_min(ll x, int a, int b, int k, int l, int r) {
      if(b <= l || r <= a || max_v[k] <= x) { return; }
      if(a <= l && r <= b && smax_v[k] < x) {
         update_node_max(k, x);
         return;
      }

      push(k);
      _update_min(x, a, b, 2 * k + 1, l, (l + r) / 2);
      _update_min(x, a, b, 2 * k + 2, (l + r) / 2, r);
      update(k);
   }

   void _update_max(ll x, int a, int b, int k, int l, int r) {
      if(b <= l || r <= a || x <= min_v[k]) { return; }
      if(a <= l && r <= b && x < smin_v[k]) {
         update_node_min(k, x);
         return;
      }

      push(k);
      _update_max(x, a, b, 2 * k + 1, l, (l + r) / 2);
      _update_max(x, a, b, 2 * k + 2, (l + r) / 2, r);
      update(k);
   }

   void addall(int k, ll x) {
      max_v[k] += x;
      if(smax_v[k] != -inf) smax_v[k] += x;
      min_v[k] += x;
      if(smin_v[k] != inf) smin_v[k] += x;

      sum[k] += len[k] * x;
      if(lval[k] != inf) {
         lval[k] += x;
      } else {
         ladd[k] += x;
      }
   }

   void updateall(int k, ll x) {
      max_v[k] = x;
      smax_v[k] = -inf;
      min_v[k] = x;
      smin_v[k] = inf;
      max_c[k] = min_c[k] = len[k];

      sum[k] = x * len[k];
      lval[k] = x;
      ladd[k] = 0;
   }

   void _add_val(ll x, int a, int b, int k, int l, int r) {
      if(b <= l || r <= a) { return; }
```

```
      if(a <= l && r <= b) {
         addall(k, x);
         return;
      }

      push(k);
      _add_val(x, a, b, 2 * k + 1, l, (l + r) / 2);
      _add_val(x, a, b, 2 * k + 2, (l + r) / 2, r);
      update(k);
   }

   void _update_val(ll x, int a, int b, int k, int l, int r) {
      if(b <= l || r <= a) { return; }
      if(a <= l && r <= b) {
         updateall(k, x);
         return;
      }

      push(k);
      _update_val(x, a, b, 2 * k + 1, l, (l + r) / 2);
      _update_val(x, a, b, 2 * k + 2, (l + r) / 2, r);
      update(k);
   }

   ll _query_max(int a, int b, int k, int l, int r) {
      if(b <= l || r <= a) { return -inf; }
      if(a <= l && r <= b) { return max_v[k]; }
      push(k);
      ll lv = _query_max(a, b, 2 * k + 1, l, (l + r) / 2);
      ll rv = _query_max(a, b, 2 * k + 2, (l + r) / 2, r);
      return max(lv, rv);
   }

   ll _query_min(int a, int b, int k, int l, int r) {
      if(b <= l || r <= a) { return inf; }
      if(a <= l && r <= b) { return min_v[k]; }
      push(k);
      ll lv = _query_min(a, b, 2 * k + 1, l, (l + r) / 2);
      ll rv = _query_min(a, b, 2 * k + 2, (l + r) / 2, r);
      return min(lv, rv);
   }

   ll _query_sum(int a, int b, int k, int l, int r) {
      if(b <= l || r <= a) { return 0; }
      if(a <= l && r <= b) { return sum[k]; }
      push(k);
      ll lv = _query_sum(a, b, 2 * k + 1, l, (l + r) / 2);
      ll rv = _query_sum(a, b, 2 * k + 2, (l + r) / 2, r);
      return lv + rv;
   }

   public:
   /// @brief セグ木を配列の内容で初期化する
   /// @param a セグ木の初期値
   ch_segtree(const vector<ll>& a) {
      n = a.size();
      n0 = 1;
      while(n0 < n) n0 <<= 1;

      max_v.resize(n0 * 2);
      smax_v.resize(n0 * 2);
      max_c.resize(n0 * 2);
      min_v.resize(n0 * 2);
      smin_v.resize(n0 * 2);
      min_c.resize(n0 * 2);
      sum.resize(n0 * 2);
      len.resize(n0 * 2);
      ladd.resize(n0 * 2);
      lval.resize(n0 * 2);

      for(int i = 0; i < 2 * n0; ++i) ladd[i] = 0, lval[i] =
inf;
      len[0] = n0;
      for(int i = 0; i < n0 - 1; ++i) len[2 * i + 1] = len[2 *
i + 2] = (len[i] >> 1);

      for(int i = 0; i < n; ++i) {
         max_v[n0 - 1 + i] = min_v[n0 - 1 + i] = sum[n0 - 1 +
i] = a[i];
         smax_v[n0 - 1 + i] = -inf;
         smin_v[n0 - 1 + i] = inf;
         max_c[n0 - 1 + i] = min_c[n0 - 1 + i] = 1;
      }

      for(int i = n; i < n0; ++i) {
         max_v[n0 - 1 + i] = smax_v[n0 - 1 + i] = -inf;
         min_v[n0 - 1 + i] = smin_v[n0 - 1 + i] = inf;
         max_c[n0 - 1 + i] = min_c[n0 - 1 + i] = 0;
      }
      for(int i = n0 - 2; i >= 0; i--) { update(i); }
   }

   /// @brief p番目の要素にアクセス
   /// @param p アクセスする要素のインデックス(0-indexed)
   /// @return p番目の要素
   ll get(int p) { return prod_sum(p, p + 1); }

   /// @brief p番目の要素をxに変更
   /// @param p 変更する要素のインデックス(0-indexed)
   /// @param x 変更後の値
   void set(int p, ll x) { apply_update(p, p + 1, x); }

   /// @brief 区間[a,b)でv[i]←min(v[i],x)
   /// @param a 区間の左端(inclusive)
   /// @param b 区間の右端(exclusive)
   /// @param x minをとる値
   void apply_chmin(int a, int b, ll x) { _update_min(x, a, b,
0, 0, n0); }

   /// @brief 区間[a,b)でv[i]←max(v[i],x)
   /// @param a 区間の左端(inclusive)
   /// @param b 区間の右端(exclusive)
   /// @param x maxをとる値
   void apply_chmax(int a, int b, ll x) { _update_max(x, a, b,
0, 0, n0); }

   /// @brief 区間[a,b)でv[i]←v[i]+x
   /// @param a 区間の左端(inclusive)
   /// @param b 区間の右端(exclusive)
   /// @param x 加算する値
   void apply_add(int a, int b, ll x) { _add_val(x, a, b, 0, 0,
n0); }

   /// @brief 区間[a,b)でv[i]←x
   /// @param a 区間の左端(inclusive)
   /// @param b 区間の右端(exclusive)
   /// @param x 更新する値
   void apply_update(int a, int b, ll x) { _update_val(x, a, b,
0, 0, n0); }

   /// @brief 区間[a,b)の最大値を取得
   /// @param a 区間の左端(inclusive)
   /// @param b 区間の右端(exclusive)
   /// @return 区間[a,b)の最大値
   ll prod_max(int a, int b) { return _query_max(a, b, 0, 0,
n0); }

   /// @brief 区間[a,b)の最小値を取得
   /// @param a 区間の左端(inclusive)
   /// @param b 区間の右端(exclusive)
   /// @return 区間[a,b)の最小値
   ll prod_min(int a, int b) { return _query_min(a, b, 0, 0,
n0); }

   /// @brief 区間[a,b)の和を取得
   /// @param a 区間の左端(inclusive)
   /// @param b 区間の右端(exclusive)
   /// @return 区間[a,b)の和
   ll prod_sum(int a, int b) { return _query_sum(a, b, 0, 0,
n0); }
};
```

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]);
   }
};
```

dsu.hpp

md5: b55e78

```
// base: d569f4
struct dsu {
   private:
   int _n;
   vector<int> p;

   public:
   dsu() : _n(0) {}
   explicit 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) return x;
      if(-p[x] < -p[y]) swap(x, 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);
      if(p[a] < 0) return a;
      int x = a;
      while(p[x] >= 0) x = p[x];
      while(p[a] >= 0) {
         int t = p[a];
         p[a] = x;
         a = t;
      }
      return x;
   }

   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;
   }  // bf3a1e
};
```

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();
  }
};
```

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)

   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]); }
};
```

sparse_table.hpp

md5: f3812e

```
template<typename T, auto op> struct sparse_table {
   vector<vector<T>> st;
   vector<int> lg;

   sparse_table(const vector<T>& v) {
      int b = 0;
      while((1 << b) <= v.size()) b++;
      st.assign(b, vector<T>(1 << b));
      for(int i = 0; i < size(v); i++) { st[0][i] = v[i]; }
      for(int i = 1; i < b; i++) {
         for(int j = 0; j + (1 << i) <= (1 << b); j++) st[i][j]
= op(st[i - 1][j], st[i - 1][j + (1 << (i - 1))]);
      }
      lg.resize(v.size() + 1);
      for(int i = 2; i < size(lg); i++) lg[i] = lg[i >> 1] + 1;
   }

   inline T prod(int l, int r) {
      int b = lg[r - l];
      return op(st[b][l], st[b][r - (1 << b)]);
   }
};
```

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;
}
```

crt.hpp

md5: 338cb4

```
// 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();
}
```

eratos.hpp

md5: 0de96f

```
void eratos(int n, vector<bool>& isprime) {
   isprime = vector<bool>(n + 1, true);
   isprime[0] = false;
   isprime[1] = false;
   int last = ((int)sqrt(n)) + 1;
   for(int i = 2; i < last; i++) {
      if(isprime[i]) {
         int j = i + i;
         while(j <= n) {
            isprime[j] = false;
            j += i;
         }
      }
   }
}
```

factorize.hpp

md5: 8e6dd4

```
// \sqrt(n)の素因数分解 a01341
map<ll, int> prime_factor(ll n) {
   map<ll, int> ret;
   for(ll i = 2; i * i <= n; i++) {
      while(n % i == 0) {
         ret[i]++;
         n /= i;
      }
   }
   if(n != 1) ret[n] = 1;
   return ret;
}

// 線形篩 1f4de3
vector<int> preprocess(int n) {
   n++;
   vector<int> res(n);
   std::iota(res.begin(), res.end(), 0);
   for(int i = 2; i * i < n; ++i) {
      if(res[i] < i) continue;
      for(int j = i * i; j < n; j += i)
         if(res[j] == j) res[j] = i;
   }
   return res;
}

map<int, int> prime_factor(int n, const vector<int>& lp) {
   map<int, int> ret;
   while(n > 1) {
      ret[lp[n]]++;
      n /= lp[n];
   }
   return ret;
}
```

floor_sum.hpp

md5: 0f7242

```
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;
}
```

modinv.hpp

md5: a0de19

```
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;
}
```

modpow.hpp

md5: 6b940c

```
ll modpow(ll a, ll n, ll MOD) {
   using T = __int128_t;
   // using T = ll;
   T res = 1;
   T mul = a;
   mul %= MOD;
   if(n < 0) {
      n = -n;
      mul = modinv(mul, MOD);
```

```
    }
    while(n > 0) {
        if(n & 1) res = (res * mul) % MOD;
        mul = (mul * mul) % MOD;
        n >>= 1;
    }
    return ll(res);
}
```

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;
}
```

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: 81b530

```
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, 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;
}
```

FFT_fast.hpp

md5: 33f77f

```
// modint を u32 にして加減算を真面目にやると速い
mm g = 3;  // 原始根
void fft(vector<mm>& a) {
   ll n = size(a), lg = __lg(n);
   static auto z = [] {
      vector<mm> z(30);
      mm s = 1;
      for(int i = 2; i < 32; i++) {
         z[i - 2] = s * g.pow(mod >> i);
         s *= g.inv().pow(mod >> i);
      }
      return z;
   }();
   for(int l = 0; l < lg; l++) {
      ll w = 1 << (lg - l - 1);
      mm s = 1;
      for(int k = 0; k < (1 << l); k++) {
         ll o = k << (lg - l);
         for(ll i = o; i < o + w; i++) {
            mm x = a[i], y = a[i + w] * s;
            a[i] = x + y;
            a[i + w] = x - y;
         }
         s *= z[countr_zero<uint64_t>(~k)];
      }
   }
}
// コピペ
void ifft(vector<mm>& a) {
   ll n = size(a), lg = __lg(n);
   static auto z = [] {
      vector<mm> z(30);
      mm s = 1;
      for(int i = 2; i < 32; i++) {  // g を逆数に
         z[i - 2] = s * g.inv().pow(mod >> i);
         s *= g.pow(mod >> i);
      }
      return z;
   }();
   for(ll l = lg; l--;) {  // 逆順に
      ll w = 1 << (lg - l - 1);
      mm s = 1;
      for(int k = 0; k < (1 << l); k++) {
         ll o = k << (lg - l);
         for(ll i = o; i < o + w; i++) {
            mm x = a[i], y = a[i + w];  // *s を下に移動
            a[i] = x + y;
            a[i + w] = (x - y) * s;
         }
         s *= z[countr_zero<uint64_t>(~k)];
      }
   }
}
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(size(a), size(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;
   ifft(a);
   a.resize(s);
   return a;
}
```

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); }
};
```

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);
      }
   }
};
```

topological_sort.hpp

md5: 024d40

```
vector<int> topological_sort(vector<vector<int>>& g) {
   int n = g.size();
   vector<int> indeg(n);
   for(int i = 0; i < n; i++)
      for(int j : g[i]) indeg[j]++;
   vector<int> res;
   queue<int> q;
   for(int i = 0; i < n; i++)
      if(indeg[i] == 0) q.push(i);
   while(!q.empty()) {
      auto v = q.front();
      q.pop();
```

```
      res.push_back(v);
      for(auto u : g[v]) {
         indeg[u]--;
         if(indeg[u] == 0) q.push(u);
      }
   }
   return res;
}
```

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;
      }
   }
};
```

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) {}
};
```

offline_lca.hpp

md5: e88371

```
vector<int> offline_lca(const vector<vector<int>>& g, const
vector<pair<int, int>>& qs, int root = 0) {
   int N = size(g);
   int Q = size(qs);
   dsu d(N);
   vector<int> mark(N), ptr(N), ans(Q, -1);
   stack<int> st;
   st.push(root);
   for(auto& [l, r] : qs) {
      mark[l]++;
      mark[r]++;
   }

   vector<vector<pair<int, int>>> q(N);
   for(int i = 0; i < N; i++) {
      q[i].reserve(mark[i]);
      mark[i] = -1;
      ptr[i] = size(g[i]);
   }
   for(int i = 0; i < Q; i++) {
      q[qs[i].first].emplace_back(qs[i].second, i);
      q[qs[i].second].emplace_back(qs[i].first, i);
   }
   auto run = [&](int u) -> bool {
      while(ptr[u]) {
         int v = g[u][--ptr[u]];
         if(mark[v] == -1) {
            st.push(v);
            return true;
         }
      }
      return false;
   };

   while(!st.empty()) {
      int u = st.top();
      if(mark[u] == -1) mark[u] = u;
      else {
         d.merge(u, g[u][ptr[u]]);
         mark[d.leader(u)] = u;
      }
      if(!run(u)) {
         for(auto& [v, i] : q[u]) {
            if(~mark[v] && ans[i] == -1) { ans[i] =
mark[d.leader(v)]; }
         }
         st.pop();
      }
   }
   return ans;
};
```

rerooting.hpp

md5: 3bb537

```
// base: b7fc0f
template<class M, bool calc_contribution = false> struct
Rerooting {
   using S = typename M::S;
   using C = typename M::C;
   vector<S> dp, memo;
   vector<vector<pair<int, C>>> g;
   map<ll, S> hash;
   int N;

   Rerooting(int n) : N(n), g(n) {}

   void add_edge(int f, int t, const C& c) {
      g[f].emplace_back(t, c);
      g[t].emplace_back(f, c);
   }

   vector<S> build() {
      memo.resize(N, M::e());
      dp.resize(N, M::e());
      dfs(0, -1);
      reroot(0, M::e());
      return dp;
   }

   private:
   void dfs(int cur, int pre = -1) {
      bool is_leaf = true;
      for(auto [to, c] : g[cur]) {
         if(to == pre) continue;
         is_leaf = false;
         dfs(to, cur);
         memo[cur] = M::merge(memo[cur], M::apply(memo[to], to,
cur, c));
      }
      if(is_leaf) { memo[cur] = M::leaf(); }
   }

   void reroot(int cur, const S val, int pre = -1) {
      vector<S> ds;
      ds.push_back(val);
      if(calc_contribution) {
         if(pre == -1) hash[cur * N + pre] = val;
      }
      for(auto [to, c] : g[cur]) {
         if(to == pre) continue;
         ds.push_back(M::apply(memo[to], to, cur, c));
         if(calc_contribution) { hash[cur * N + to] =
ds.back(); }
      }
      int n = size(ds);
      vector<S> l(n + 1, M::e()), r(n + 1, M::e());
      for(int i = 0; i < n; i++) l[i + 1] = M::merge(l[i],
ds[i]);
      for(int i = n - 1; i >= 0; i--) r[i] = M::merge(ds[i],
r[i + 1]);
      dp[cur] = r[0];
      int ind = 1;
      for(auto [to, c] : g[cur]) {
         if(to == pre) continue;
         S sub = M::merge(l[ind], r[ind + 1]);
         reroot(to, M::apply(sub, cur, to, c), cur);
         ind++;
      }
   }

   public:
   S get_contribution(int p, int c) {
      if(hash.count(p * N + c)) return hash[p * N + c];
      return M::e();
   }  // e6000f
};

/*

struct M {
   using S = pair<mm, int>; // DPの型
   using C = pair<mm, mm>;  // 辺コストの型
   static S merge(S a, S b) { return {a.first + b.first,
a.second + b.second}; } // DPのマージ
   static S apply(S a, int from, int to, C b) {  // DPの親への寄
与
      return {(a.first + A[from]) * b.first + b.second *
(a.second + 1), a.second + 1};
   }
   static S e() { return {0, 0}; } // 単位元
   static S leaf() { return {0, 0}; } // 葉の値
};

Rerooting<M> reroot;
*/
```

flow

燃やす埋める.md

変形前の制約	変形後の制約
x が 0 のとき z 失う	(x, T, z)
x が 0 のとき z 得る	無条件で z 得る; (S, x, z)

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

string

KMP.hpp

md5: 298f79

```cpp
// 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

```cpp
// 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

```cpp
// 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

```cpp
// 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

```cpp
// 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

```cpp
// 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

doubling.hpp

md5: df858f

```
template<int L> struct Doubling {
   private:
   vector<vector<int>> V;

   public:
   Doubling(const vector<int>& v) {
      int N = size(v);
      V = vector<vector<int>>(L, vector<int>(N));
      for(int i = 0; i < N; i++) V[0][i] = v[i];
      for(int i = 0; i < L - 1; i++)
         for(int j = 0; j < N; j++) {
            if(V[i][j] != -1) V[i + 1][j] = V[i][V[i][j]];
         }
   }

   int jump(int from, ll k) {
      for(int cnt = 0; k > 0; k >>= 1, ++cnt) {
         if((k & 1) && from != -1) from = V[cnt][from];
      }
      return from;
   }
};
```

doubling_monoid.hpp

md5: 530e69

```
template<int L, class T, auto op, auto e> struct Doubling {
   private:
   vector<vector<int>> V;
   vector<vector<T>> data;

   public:
   Doubling(const vector<int>& to, const vector<T>& v) {
      int N = size(to);
      V = vector<vector<int>>(L, vector<int>(N, -1));
      data = vector<vector<T>>(L, vector<T>(N, e()));
      for(int i = 0; i < N; i++) {
         V[0][i] = to[i];
         data[0][i] = v[i];
      }

      for(int i = 0; i < L - 1; i++) {
         for(int j = 0; j < N; j++) {
            if(V[i][j] != -1) {
               V[i + 1][j] = V[i][V[i][j]];
               data[i + 1][j] = op(data[i][j], data[i][V[i]
[j]]);
            } else {
               V[i + 1][j] = V[i][j];
               data[i + 1][j] = data[i][j];
            }
         }
      }
   }

   pair<int, T> jump(int from, ll k) {
      T res = e();
      for(int cnt = 0; k > 0; k >>= 1, ++cnt) {
         if((k & 1) && from != -1) {
            res = op(res, data[cnt][from]);
            from = V[cnt][from];
         }
      }
      return {from, res};
   }
};
```

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

素数の個数

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

高度合成数

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

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

素数階乗

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

階乗

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

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;
}
```

priority_queue.md

	push	pop	modify	erase
std::priority_queue	最悪$\Theta(n)$, 償却$\Theta(\log(n))$	最悪$\Theta(\log(n))$	最悪$\Theta(n\log(n))$	$\Theta(n\log(n))$
pairing_heap_tag	$O(1)$	最悪$\Theta(n)$, 償却$\Theta(\log n)$	最悪$\Theta(n)$, 償却$\Theta(\log(n))$	最悪$\Theta(n)$, 償却$O(\log(n))$
binary_heap_tag	最悪$\Theta(n)$, 償却$O(\log(n))$	最悪$\Theta(n)$, 償却$O(\log(n))$	$\Theta(n)$	$\Theta(n)$

thin_heap_tag	$O(1)$	最悪$\Theta(n)$, 償却$O(\log(n))$	最悪$\Theta(\log(n))$,償却$O(1)$ or 償却$\Theta(\log n)$	最悪$\Theta(n)$, 償却$O(\log(n))$

joinは基本的にeraseの最悪計算量に一致.pairing_heap_tagのみ高速で$O(1)$