ACM ICPC Team Reference - NTHU SplayTreap

Contents

1 Basic

1.1 vimrc

```
se nu ai hls et ru ic is sc cul
se re=1 ts=4 sts=4 sw=4 ls=2 mouse=a
syntax on
hi cursorline cterm=none ctermbg=89
set bg=dark
inoremap {<CR> {<CR>}<Esc>ko<tab>
```

2 Data-Structure

2.1 CDQ

```cpp
void CDQ(int l, int r) {
  if(l + 1 == r) return;
  int mid = (l + r) / 2;
  CDQ(l, mid), CDQ(mid, r);
  int i = l;
  for(int j = mid; j < r; j++) {
    const Q& q = qry[j];
    while(i < mid && qry[i].x >= q.x) {
      if(qry[i].id == -1) fenw.add(qry[i].y, qry[i].w);
      i++;
    }
    if(q.id >= 0) ans[q.id] += q.w * fenw.sum(q.y, sz - 1);
  }
  for(int p = l; p < i; p++) if(qry[p].id == -1) fenw.add(qry[p].y, -qry[p].w);
  inplace_merge(qry.begin() + l, qry.begin() + mid, qry.begin() + r, [](const Q& a, const Q& b) {
    return a.x > b.x;
  });
}
```

2.2 CHT

```cpp
struct line_t {
  mutable ll k, m, p;
  bool operator<(const line_t& o) const { return k < o.k; }
  bool operator<(ll x) const { return p < x; }
};
template<bool MAX>
struct CHT : multiset<line_t, less<>> {
  const ll INF = 1e18L;
  bool isect(iterator x, iterator y) {
    if(y == end()) return x->p = INF, 0;
    if(x->k == y->k) {
      x->p = (x->m > y->m ? INF : -INF);
    } else {
      x->p = floor_div(y->m - x->m, x->k - y->k); // see Math
    }
    return x->p >= y->p;
  }
  void add_line(ll k, ll m) {
    if(!MAX) k = -k, m = -m;
    auto z = insert({k, m, 0}), y = z++, x = y;
    while(isect(y, z)) z = erase(z);
    if(x != begin() && isect(--x, y)) isect(x, y = erase(y));
    while((y = x) != begin() && (--x)->p >= y->p) isect(x, erase(y));
  }
  ll get(ll x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return (l.k * x + l.m) * (MAX ? +1 : -1);
  }
};
```

2.3 DLX

```cpp
struct DLX {
  int n, m, tot, ans;
  vi first, siz, L, R, U, D, col, row, stk;
  DLX(int _n, int _m) : n(_n), m(_m), tot(_m) {
    int sz = n * m;
    first = siz = L = R = U = D = col = row = stk = vi(sz);
    REP(i, m + 1) {
      L[i] = i - 1, R[i] = i + 1;
      U[i] = D[i] = i;
    }
    L[0] = m, R[m] = 0;
  }
  void insert(int r, int c) {
    r++, c++;
    col[++tot] = c, row[tot] = r, ++siz[c];
    D[tot] = D[c], U[D[c]] = tot, U[tot] = c, D[c] = tot;
    if(!first[r]) first[r] = L[tot] = R[tot] = tot;
    else {
      L[R[tot] = R[first[r]]] = tot;
      R[L[tot] = first[r]] = tot;
    }
  }
  #define TRAV(i, X, j) for(i = X[j]; i != j; i = X[i])
  void remove(int c) {
    int i, j;
    L[R[c]] = L[c], R[L[c]] = R[c];
    TRAV(i, D, c) TRAV(j, R, i) {
      D[U[D[j]] = U[j]] = D[j];
      siz[col[j]]--;
    }
  }
  void recover(int c) {
    int i, j;
    TRAV(i, U, c) TRAV(j, L, i) {
      U[D[j]] = D[U[j]] = j;
      siz[col[j]]++;
    }
    L[R[c]] = R[L[c]] = c;
  }
  bool dance(int dep) {
    if(!R[0]) return ans = dep, true;
    int i, j, c = R[0];
    TRAV(i, R, 0) if(siz[i] < siz[c]) c = i;
    remove(c);
    TRAV(i, D, c) {
      stk[dep] = row[i];
      TRAV(j, R, i) remove(col[j]);
      if(dance(dep + 1)) return true;
      TRAV(j, L, i) recover(col[j]);
    }
    recover(c);
    return false;
  }
  vi solve() {
    if(!dance(1)) return {};
    return vi(stk.begin() + 1, stk.begin() + ans);
  }
};
```

2.4 fast-set

```cpp
// Can correctly work with numbers in range [0; MAXN]
// Supports all std::set operations in O(1) on random queries / dense arrays, O(log_64(N)) in worst case (sparce array).
// Count operation works in O(1) always.
template<uint MAXN>
class fast_set {
private:
  static const uint PREF = (MAXN <= 64 ? 0 :
                MAXN <= 4096 ? 1 :
                MAXN <= 262144 ? 1 + 64 :
                MAXN <= 16777216 ? 1 + 64 + 4096 :
                MAXN <= 1073741824 ? 1 + 64 + 4096 + 262144 : 227) + 1;
  static constexpr ull lb(int x) {
    if(x == 64) return ULLONG_MAX;
    return (1ULL << x) - 1;
  };
  static const uint SZ = PREF + (MAXN + 63) / 64 + 1;
  ull m[SZ] = {0};
  inline uint left(uint v) const { return (v - 62) * 64; }
  inline uint parent(uint v) const { return v / 64 + 62; }
  inline void setbit(uint v) { m[v >> 6] |= 1ULL << (v & 63); }
  inline void resetbit(uint v) { m[v >> 6] &= ~(1ULL << (v & 63)); }
  inline uint getbit(uint v) const { return m[v >> 6] >> (v & 63) & 1; }
  inline ull childs_value(uint v) const { return m[left(v) >> 6]; }
  inline int left_go(uint x, const uint c) const {
    const ull rem = x & 63;
    uint bt = PREF * 64 + x;
    ull num = m[bt >> 6] & lb(rem + c);
    if(num) return (x ^ rem) | __lg(num);
    for(bt = parent(bt); bt > 62; bt = parent(bt)) {
      const ull rem = bt & 63;
      num = m[bt >> 6] & lb(rem);
      if(num) {
        bt = (bt ^ rem) | __lg(num);
        break;
      }
    }
    if(bt == 62) return -1;
    while(bt < PREF * 64) bt = left(bt) | __lg(m[bt - 62]);
    return bt - PREF * 64;
  }
  inline int right_go(uint x, const uint c) const {
    const ull rem = x & 63;
    uint bt = PREF * 64 + x;
    ull num = m[bt >> 6] & ~lb(rem + c);
    if(num) return (x ^ rem) | __builtin_ctzll(num);
    for(bt = parent(bt); bt > 62; bt = parent(bt)) {
      const ull rem = bt & 63;
      num = m[bt >> 6] & ~lb(rem + 1);
      if(num) {
        bt = (bt ^ rem) | __builtin_ctzll(num);
        break;
      }
    }
    if(bt == 62) return -1;
    while(bt < PREF * 64) bt = left(bt) | __builtin_ctzll(m[bt - 62]);
    return bt - PREF * 64;
  }
public:
  fast_set() { assert(PREF != 228); setbit(62); }
  bool empty() const {return getbit(63);}
  void clear() { fill(m, m + SZ, 0); setbit(62); }
  bool count(uint x) const { return m[PREF + (x >> 6)] >> (x & 63) & 1; }
  void insert(uint x) { for(uint v = PREF * 64 + x; !getbit(v); v = parent(v)) setbit(v); }
  void erase(uint x) {
    if(!getbit(PREF * 64 + x)) return;
    resetbit(PREF * 64 + x);
    for(uint v = parent(PREF * 64 + x); v > 62 && !childs_value(v); v = parent(v)) resetbit(v);
  }
```

```
  int find_next(uint x) const { return right_go(x, 0); } // >=
  int find_prev(uint x) const { return left_go(x, 1); } // <=
};
```

2.5 lazysegtree

```
template<class S,
         S (*e)(),
         S (*op)(S, S),
         class F,
         F (*id)(),
         S (*mapping)(F, S),
         F (*composition)(F, F)>
struct lazy_segtree {
  int n, size, log;
  vector<S> d; vector<F> lz;
  void update(int k) { d[k] = op(d[k << 1], d[k << 1 | 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(k << 1, lz[k]);
    all_apply(k << 1 | 1, lz[k]);
    lz[k] = id();
  }
  lazy_segtree(int _n) : lazy_segtree(vector<S>(_n, e())) {}
  lazy_segtree(const vector<S>& v) : n(sz(v)) {
    log = __lg(2 * n - 1), size = 1 << log;
    d.resize(size * 2, e());
    lz.resize(size, id());
    REP(i, n) d[size + i] = v[i];
    for(int i = size - 1; i; i--) update(i);
  }
  void set(int p, S x) {
    p += size;
    for(int i = log; i; --i) push(p >> i);
    d[p] = x;
    for(int i = 1; i <= log; ++i) update(p >> i);
  }
  S get(int p) {
    p += size;
    for(int i = log; i; i--) push(p >> i);
    return d[p];
  }
  S prod(int l, int r) {
    if(l == r) return e();
    l += size; r += size;
    for(int i = log; i; i--) {
      if(((l >> i) << i) != l) push(l >> i);
      if(((r >> i) << i) != r) push(r >> i);
    }
    S sml = e(), smr = e();
    while(l < r) {
      if(l & 1) sml = op(sml, d[l++]);
      if(r & 1) smr = op(d[--r], smr);
      l >>= 1;
      r >>= 1;
    }
    return op(sml, smr);
  }
  S all_prod() const { return d[1]; }
  void apply(int p, F f) {
    p += size;
    for(int i = log; i; i--) push(p >> i);
    d[p] = mapping(f, d[p]);
    for(int i = 1; i <= log; i++) update(p >> i);
  }
  void apply(int l, int r, F f) {
    if(l == r) return;
    l += size; r += size;
    for(int i = log; i; 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 && g(e()));
    if(l == n) return n;
    l += size;
    for(int i = log; i; i--) push(l >> i);
    S sm = e();
    do {
      while(!(l & 1)) l >>= 1;
      if(!g(op(sm, d[l]))) {
        while(l < size) {
          push(l);
          l <<= 1;
          if(g(op(sm, d[l]))) sm = op(sm, d[l++]);
        }
        return l - size;
      }
      sm = op(sm, d[l++]);
    } while((l & -l) != l);
    return n;
  }
  template<class G> int min_left(int r, G g) {
    assert(0 <= r && r <= n && 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 & 1)) r >>= 1;
      if(!g(op(d[r], sm))) {
        while(r < size) {
          push(r);
          r = r << 1 | 1;
          if(g(op(d[r], sm))) sm = op(d[r--], sm);
        }
        return r + 1 - size;
      }
      sm = op(d[r], sm);
    } while((r & -r) != r);
    return 0;
  }
};
```

2.6 LCT

```
template<class S,
         S (*e)(),
         S (*op)(S, S),
         S (*reversal)(S),
         class F,
         F (*id)(),
         S (*mapping)(F, S),
         F (*composition)(F, F)>
struct lazy_lct {
  struct Node {
    S val = e(), sum = e();
    F lz = id();
    bool rev = false;
    int sz = 1;
    Node *l = nullptr, *r = nullptr, *p = nullptr;
    Node() {}
    Node(const S& s) : val(s), sum(s) {}
    bool is_root() const { return p == nullptr || (p->l != this && p->r != this); }
  };
  int n;
  vector<Node> a;
  lazy_lct() : n(0) {}
  explicit lazy_lct(int _n) : lazy_lct(vector<S>(_n, e())) {}
  explicit lazy_lct(const vector<S>& v) : n(SZ(v)) { REP(i, n) a.eb(v[i]); }
  Node* access(int u) {
    Node* v = &a[u];
    Node* last = nullptr;
    for(Node* p = v; p != nullptr; p = p->p) splay(p), p->r = last, pull(last = p);
    splay(v);
    return last;
  }
  void make_root(int u) { access(u), a[u].rev ^= 1, push(&a[u]); }
  void link(int u, int v) { make_root(v), a[v].p = &a[u]; }
  void cut(int u) {
    access(u);
    if(a[u].l != nullptr) a[u].l->p = nullptr, a[u].l = nullptr, pull(&a[u]);
  }
  void cut(int u, int v) { make_root(u), cut(v); }
  bool is_connected(int u, int v) {
    if(u == v) return true;
    return access(u), access(v), a[u].p != nullptr;
  }
  int get_lca(int u, int v) { return access(u), access(v) - &a[0]; }
  void set(int u, const S& s) { access(u), a[u].val = s, pull(&a[u]); }
  S get(int u) { return access(u), a[u].val; }
  void apply(int u, int v, const F& f) { make_root(u), access(v), all_apply(&a[v], f), push(&a[v]); }
  S prod(int u, int v) { return make_root(u), access(v), a[v].sum; }
  void rotate(Node* v) {
    auto attach = [&](Node* p, bool side, Node* c) {
      (side ? p->r : p->l) = c;
      pull(p);
      if(c != nullptr) c->p = p;
    };
    Node *p = v->p, *g = p->p;
    bool rgt = (p->r == v);
    bool rt = p->is_root();
    attach(p, rgt, (rgt ? v->l : v->r));
    attach(v, !rgt, p);
    if(!rt) attach(g, (g->r == p), v);
    else v->p = g;
  }
  void splay(Node* v) {
    push(v);
    while(!v->is_root()) {
      auto p = v->p;
      auto g = p->p;
      if(!p->is_root()) push(g);
      push(p), push(v);
      if(!p->is_root()) rotate((g->r == p) == (p->r == v) ? p : v);
      rotate(v);
    }
  }
  void all_apply(Node* v, F f) {
    v->val = mapping(f, v->val), v->sum = mapping(f, v->sum);
    v->lz = composition(f, v->lz);
  }
  void push(Node* v) {
    if(v->lz != id()) {
      if(v->l != nullptr) all_apply(v->l, v->lz);
      if(v->r != nullptr) all_apply(v->r, v->lz);
      v->lz = id();
    }
```

```cpp
    if(v->rev) {
      swap(v->l, v->r);
      if(v->l != nullptr) v->l->rev ^= 1;
      if(v->r != nullptr) v->r->rev ^= 1;
      v->sum = reversal(v->sum);
      v->rev = false;
    }
  }
  void pull(Node* v) {
    v->sz = 1;
    v->sum = v->val;
    if(v->l != nullptr) {
      push(v->l);
      v->sum = op(v->l->sum, v->sum);
      v->sz += v->l->sz;
    }
    if(v->r != nullptr) {
      push(v->r);
      v->sum = op(v->sum, v->r->sum);
      v->sz += v->r->sz;
    }
  }
};
```

2.7 LiChao

```cpp
template<class T>
struct LiChao {
  static constexpr T INF = numeric_limits<T>::max();
  struct Line {
    T a, b;
    Line(T a, T b) : a(a), b(b) {}
    T operator()(T x) const { return a * x + b; }
  };
  int n;
  vector<Line> fs;
  vector<T> xs;
  LiChao(const vector<T>& xs_) {
    xs = sort_unique(xs_);
    n = SZ(xs);
    fs.assign(2 * n, Line(T(0), INF));
  }
  int index(T x) const { return lower_bound(ALL(xs), x) - xs.begin(); }
  void add_line(T a, T b) { update(a, b, 0, n); }
  // [xl, xr) ax+b
  void add_segment(T xl, T xr, T a, T b) {
    int l = index(xl), r = index(xr);
    update(a, b, l, r);
  }
  void update(T a, T b, int l, int r) {
    Line g(a, b);
    for(l += n, r += n; l < r; l >>= 1, r >>= 1){
      if(l & 1) descend(g, l++);
      if(r & 1) descend(g, --r);
    }
  }
  void descend(Line g, int i) {
    int l = i, r = i + 1;
    while(l < n) l <<= 1, r <<= 1;
    while(l < r) {
      int c = (l + r) / 2;
      T xl = xs[l - n], xr = xs[r - 1 - n], xc = xs[c - n];
      Line& f = fs[i];
      if(f(xl) <= g(xl) && f(xr) <= g(xr)) return;
      if(f(xl) >= g(xl) && f(xr) >= g(xr)) { f = g; return; }
      if(f(xc) > g(xc)) swap(f, g);
      if(f(xl) > g(xl)) i = 2 * i, r = c;
      else i = 2 * i + 1, l = c;
    }
  }
  T get(T x) {
    int i = index(x);
    T res = INF;
    for(i += n; i; i >>= 1) res = min(res, fs[i](x));
    return res;
  }
};
```

2.8 rect-add-rect-sum

```cpp
template<class Int, class T>
struct RectangleAddRectangleSum {
  struct AQ { Int xl, xr, yl, yr; T val; };
  struct SQ { Int xl, xr, yl, yr; };
  vector<AQ> add_qry;
  vector<SQ> sum_qry;
  // A[x][y] += val for(x, y) in [xl, xr) * [yl, yr)
  void add_rectangle(Int xl, Int xr, Int yl, Int yr, T val) { add_qry.pb({xl, xr, yl, yr, val}); }
  // Get sum of A[x][y] for(x, y) in [xl, xr) * [yl, yr)
  void add_query(Int xl, Int xr, Int yl, Int yr) { sum_qry.pb({xl, xr, yl, yr}); }
  vector<T> solve() {
    vector<Int> ys;
    for(auto &a : add_qry) ys.pb(a.yl), ys.pb(a.yr);
    ys = sort_unique(ys);
    const int Y = SZ(ys);
    vector<tuple<Int, int, int>> ops;
    REP(q, SZ(sum_qry)) {
      ops.eb(sum_qry[q].xl, 0, q);
      ops.eb(sum_qry[q].xr, 1, q);
    }
    REP(q, SZ(add_qry)) {
      ops.eb(add_qry[q].xl, 2, q);
      ops.eb(add_qry[q].xr, 3, q);
    }
    sort(ALL(ops));
    fenwick<T> b00(Y), b01(Y), b10(Y), b11(Y);
    vector<T> ret(SZ(sum_qry));
    for(auto o : ops) {
      int qtype = get<1>(o), q = get<2>(o);
      if(qtype >= 2) {
        const auto& query = add_qry[q];
        int i = lower_bound(ALL(ys), query.yl) - ys.begin();
        int j = lower_bound(ALL(ys), query.yr) - ys.begin();
        T x = get<0>(o);
        T yi = query.yl, yj = query.yr;
        if(qtype & 1) swap(i, j), swap(yi, yj);
        b00.add(i, x * yi * query.val);
        b01.add(i, -x * query.val);
        b10.add(i, -yi * query.val);
        b11.add(i, query.val);
        b00.add(j, -x * yj * query.val);
        b01.add(j, x * query.val);
        b10.add(j, yj * query.val);
        b11.add(j, -query.val);
      } else {
        const auto& query = sum_qry[q];
        int i = lower_bound(ALL(ys), query.yl) - ys.begin();
        int j = lower_bound(ALL(ys), query.yr) - ys.begin();
        T x = get<0>(o);
        T yi = query.yl, yj = query.yr;
        if(qtype & 1) swap(i, j), swap(yi, yj);
        ret[q] += b00.get(i - 1) + b01.get(i - 1) * yi + b10.get(i - 1) * x + b11.get(i - 1) * x * yi;
        ret[q] -= b00.get(j - 1) + b01.get(j - 1) * yj + b10.get(j - 1) * x + b11.get(j - 1) * x * yj;
      }
    }
    return ret;
  }
};
```

2.9 rollback-dsu

```cpp
struct RollbackDSU {
  int n; vi sz, tag;
  vector<tuple<int, int, int, int>> op;
  void init(int _n) {
    n = _n;
    sz.assign(n, -1);
    tag.clear();
  }
  int leader(int x) {
    while(sz[x] >= 0) x = sz[x];
    return x;
  }
  bool merge(int x, int y) {
    x = leader(x), y = leader(y);
    if(x == y) return false;
    if(-sz[x] < -sz[y]) swap(x, y);
    op.eb(x, sz[x], y, sz[y]);
    sz[x] += sz[y]; sz[y] = x;
    return true;
  }
  int size(int x) { return -sz[leader(x);] }
  void add_tag() { tag.pb(sz(op)); }
  void rollback() {
    int z = tag.back(); tag.ppb();
    while(sz(op) > z) {
      auto [x, sx, y, sy] = op.back(); op.ppb();
      sz[x] = sx;
      sz[y] = sy;
    }
  }
};
```

2.10 segtree-beats

```cpp
struct segtree_beats {
  static constexpr ll INF = numeric_limits<ll>::max() / 2.1;
  struct alignas(32) Node {
    ll sum = 0, g1 = 0, l1 = 0;
    ll g2 = -INF, gc = 1, l2 = INF, lc = 1, add = 0;
  };
  ll n, log;
  vector<Node> v;
  segtree_beats() {}
  segtree_beats(int _n) : segtree_beats(vector<ll>(_n)) {}
  segtree_beats(const vector<ll>& vc) {
    n = 1, log = 0;
    while(n < SZ(vc)) n <<= 1, log++;
    v.resize(2 * n);
    REP(i, SZ(vc)) v[i + n].sum = v[i + n].g1 = v[i + n].l1 = vc[i];
    for(ll i = n - 1; i; --i) update(i);
  }
  void range_chmin(int l, int r, ll x) { inner_apply<1>(l, r, x); }
  void range_chmax(int l, int r, ll x) { inner_apply<2>(l, r, x); }
  void range_add(int l, int r, ll x) { inner_apply<3>(l, r, x); }
  void range_update(int l, int r, ll x) { inner_apply<4>(l, r, x); }
  ll range_min(int l, int r) { return inner_fold<1>(l, r); }
  ll range_max(int l, int r) { return inner_fold<2>(l, r); }
  ll range_sum(int l, int r) { return inner_fold<3>(l, r); }
  void update(int k) {
    Node& p = v[k];
    Node& l = v[k * 2];
    Node& r = v[k * 2 + 1];
    p.sum = l.sum + r.sum;
    if(l.g1 == r.g1) {
      p.g1 = l.g1;
      p.g2 = max(l.g2, r.g2);
      p.gc = l.gc + r.gc;
    } else {
      bool f = l.g1 > r.g1;
      p.g1 = f ? l.g1 : r.g1;
      p.gc = f ? l.gc : r.gc;
      p.g2 = max(f ? r.g1 : l.g1, f ? l.g2 : r.g2);
```

```
    }
    if(l.l1 == r.l1) {
      p.l1 = l.l1;
      p.l2 = min(l.l2, r.l2);
      p.lc = l.lc + r.lc;
    } else {
      bool f = l.l1 < r.l1;
      p.l1 = f ? l.l1 : r.l1;
      p.lc = f ? l.lc : r.lc;
      p.l2 = min(f ? r.l1 : l.l1, f ? l.l2 : r.l2);
    }
  }
  void push_add(int k, ll x) {
    Node& p = v[k];
    p.sum += x << (log + __builtin_clz(k) - 31);
    p.g1 += x, p.l1 += x;
    if(p.g2 != -INF) p.g2 += x;
    if(p.l2 != INF) p.l2 += x;
    p.add += x;
  }
  void push_min(int k, ll x) {
    Node& p = v[k];
    p.sum += (x - p.g1) * p.gc;
    if(p.l1 == p.g1) p.l1 = x;
    if(p.l2 == p.g1) p.l2 = x;
    p.g1 = x;
  }
  void push_max(int k, ll x) {
    Node& p = v[k];
    p.sum += (x - p.l1) * p.lc;
    if(p.g1 == p.l1) p.g1 = x;
    if(p.g2 == p.l1) p.g2 = x;
    p.l1 = x;
  }
  void push(int k) {
    Node& p = v[k];
    if(p.add != 0) {
      push_add(k * 2, p.add);
      push_add(k * 2 + 1, p.add);
      p.add = 0;
    }
    if(p.g1 < v[k * 2].g1) push_min(k * 2, p.g1);
    if(p.l1 > v[k * 2].l1) push_max(k * 2, p.l1);
    if(p.g1 < v[k * 2 + 1].g1) push_min(k * 2 + 1, p.g1);
    if(p.l1 > v[k * 2 + 1].l1) push_max(k * 2 + 1, p.l1);
  }

  void subtree_chmin(int k, ll x) {
    if(v[k].g1 <= x) return;
    if(v[k].g2 < x) {
      push_min(k, x);
      return;
    }
    push(k);
    subtree_chmin(k * 2, x), subtree_chmin(k * 2 + 1, x);
    update(k);
  }

  void subtree_chmax(int k, ll x) {
    if(x <= v[k].l1) return;
    if(x < v[k].l2) {
      push_max(k, x);
      return;
    }
    push(k);
    subtree_chmax(k * 2, x), subtree_chmax(k * 2 + 1, x);
    update(k);
  }
  template<int cmd>
  inline void _apply(int k, ll x) {
    if constexpr(cmd == 1) subtree_chmin(k, x);
    if constexpr(cmd == 2) subtree_chmax(k, x);
    if constexpr(cmd == 3) push_add(k, x);
    if constexpr(cmd == 4) subtree_chmin(k, x), subtree_chmax(k, x);
  }
  template<int cmd>
  void inner_apply(int l, int r, ll x) {
    if(l == r) return;
    l += n, r += n;
    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) _apply<cmd>(l++, x);
        if(r & 1) _apply<cmd>(--r, x);
        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<int cmd>
  inline ll e() {
    if constexpr(cmd == 1) return INF;
    if constexpr(cmd == 2) return -INF;
    return 0;
  }
  template<int cmd>
  inline void op(ll& a, const Node& b) {
    if constexpr(cmd == 1) a = min(a, b.l1);
    if constexpr(cmd == 2) a = max(a, b.g1);
    if constexpr(cmd == 3) a += b.sum;
  }
  template<int cmd>
  ll inner_fold(int l, int r) {
    if(l == r) return e<cmd>();
    l += n, r += n;
    for(int i = log; i >= 1; i--) {
      if(((l >> i) << i) != l) push(l >> i);
      if(((r >> i) << i) != r) push((r - 1) >> i);
    }
    ll lx = e<cmd>(), rx = e<cmd>();
    while (l < r) {
      if(l & 1) op<cmd>(lx, v[l++]);
      if(r & 1) op<cmd>(rx, v[--r]);
      l >>= 1, r >>= 1;
    }
    if constexpr(cmd == 1) lx = min(lx, rx);
    if constexpr(cmd == 2) lx = max(lx, rx);
    if constexpr(cmd == 3) lx += rx;
    return lx;
  }
};
```

2.11 segtree

```
template<class S, S (*e)(), S (*op)(S, S)>
struct segtree {
  int n, size, log;
  vector<S> st;
  void update(int v) { st[v] = op(st[v << 1], st[v << 1 | 1]); }
  segtree(int _n) : segtree(vector<S>(_n, e())) {}
  segtree(const vector<S>& a): n(sz(a)) {
    log = __lg(2 * n - 1), size = 1 << log;
    st.resize(size << 1, e());
    REP(i, n) st[size + i] = a[i];
    for(int i = size - 1; i; i--) update(i);
  }
  void set(int p, S val) {
    st[p += size] = val;
    for(int i = 1; i <= log; ++i) update(p >> i);
  }
  S get(int p) const {
    return st[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, st[l++]);
      if(r & 1) smr = op(st[--r], smr);
      l >>= 1;
      r >>= 1;
    }
    return op(sml, smr);
  }
  S all_prod() const { return st[1]; }
  template<class F> int max_right(int l, F f) const {
    assert(0 <= l && l <= n && f(e()));
    if(l == n) return n;
    l += size;
    S sm = e();
    do {
      while(~l & 1) l >>= 1;
      if(!f(op(sm, st[l]))) {
        while(l < size) {
          l <<= 1;
          if(f(op(sm, st[l]))) sm = op(sm, st[l++]);
        }
        return l - size;
      }
      sm = op(sm, st[l++]);
    } while((l & -l) != l);
    return n;
  }
  template<class F> int min_left(int r, F f) const {
    assert(0 <= r && r <= n && f(e()));
    if(r == 0) return 0;
    r += size;
    S sm = e();
    do {
      r--;
      while(r > 1 && (r & 1)) r >>= 1;
      if(!f(op(st[r], sm))) {
        while(r < size) {
          r = r << 1 | 1;
          if(f(op(st[r], sm))) sm = op(st[r--], sm);
        }
        return r + 1 - size;
      }
      sm = op(st[r], sm);
    } while((r & -r) != r);
    return 0;
  }
};
```

2.12 sparse-table

```
template<class T, T (*op)(T, T)>
struct sparse_table {
  int n;
  vector<vector<T>> b;
  sparse_table(const vector<T>& a) : n(sz(a)) {
    int lg = __lg(n) + 1;
    b.resize(lg); b[0] = a;
    for(int j = 1; j < lg; ++j) {
      b[j].resize(n - (1 << j) + 1);
      REP(i, n - (1 << j) + 1) b[j][i] = op(b[j - 1][i], b[j - 1][i + (1 << (j - 1))]);
    }
  }
  T prod(int from, int to) {
    int lg = __lg(to - from + 1);
    return op(b[lg][from], b[lg][to - (1 << lg) + 1]);
  }
};
```

2.13 static-range-inversion

```
struct static_range_inversion {
  int sz;
```

```
  vi a, L, R;
  vector<ll> ans;
  static_range_inversion(vi _a) : a(_a) {
    _a = sort_unique(_a);
    REP(i, SZ(a)) a[i] = lower_bound(ALL(_a), a[i]) - _a.begin();
    sz = SZ(_a);
  }
  void add_query(int l, int r) { L.push_back(l), R.push_back(r); }
  vector<ll> solve() {
    const int q = SZ(L);
    const int B = max(1.0, SZ(a) / sqrt(q));
    vi ord(q);
    iota(ALL(ord), 0);
    sort(ALL(ord), [&](int i, int j) {
      if(L[i] / B == L[j] / B) {
        return L[i] / B & 1 ? R[i] > R[j] : R[i] < R[j];
      }
      return L[i] < L[j];
    });
    ans.resize(q);
    fenwick<ll> fenw(sz + 1);
    ll cnt = 0;
    auto AL = [&](int i) {
      cnt += fenw.sum(0, a[i] - 1);
      fenw.add(a[i], +1);
    };
    auto AR = [&](int i) {
      cnt += fenw.sum(a[i] + 1, sz);
      fenw.add(a[i], +1);
    };
    auto DL = [&](int i) {
      cnt -= fenw.sum(0, a[i] - 1);
      fenw.add(a[i], -1);
    };
    auto DR = [&](int i) {
      cnt -= fenw.sum(a[i] + 1, sz);
      fenw.add(a[i], -1);
    };
    int l = 0, r = 0;
    for(int i = 0; i < q; i++) {
      int id = ord[i], ql = L[id], qr = R[id];
      while(l > ql) AL(--l);
      while(r < qr) AR(r++);
      while(l < ql) DL(l++);
      while(r > qr) DR(--r);
      ans[id] = cnt;
    }
    return ans;
  }
};
```

2.14 static-range-lis

```
#define MEM(a, x, n) memset(a, x, sizeof(int) * n)
using I = int*;
struct static_range_lis {
  int n, ps = 0;
  I invp, res_monge, pool;
  vector<vector<pii>> qry;
  vi ans;
  static_range_lis(vi a) : n(SZ(a)), qry(n + 1) {
    // a must be permutation of [0, n)
    pool = (I) malloc(sizeof(int) * n * 100);
    invp = A(n), res_monge = A(n);
    REP(i, n) invp[a[i]] = i;
  }
  inline I A(int x) { return pool + (ps += x) - x; }
  void add_query(int l, int r) { qry[l].pb({r, SZ(ans)}), ans.pb(r - l); }
  void unit_monge_mult(I a, I b, I r, int n) {
    if(n == 2){
      if(!a[0] && !b[0]) r[0] = 0, r[1] = 1;
      else r[0] = 1, r[1] = 0;
      return;
    }
    if(n == 1) return r[0] = 0, void();
    int lps = ps, d = n / 2;
    I a1 = A(d), a2 = A(n - d), b1 = A(d), b2 = A(n - d);
    I mpa1 = A(d), mpa2 = A(n - d), mpb1 = A(d), mpb2 = A(n - d);
    int p[2] = {};
    REP(i, n) {
      if(a[i] < d) a1[p[0]] = a[i], mpa1[p[0]++] = i;
      else a2[p[1]] = a[i] - d, mpa2[p[1]++] = i;
    }
    p[0] = p[1] = 0;
    REP(i, n) {
      if(b[i] < d) b1[p[0]] = b[i], mpb1[p[0]++] = i;
      else b2[p[1]] = b[i] - d, mpb2[p[1]++] = i;
    }
    I c1 = A(d), c2 = A(n - d);
    unit_monge_mult(a1, b1, c1, d), unit_monge_mult(a2, b2, c2, n - d);
    I cpx = A(n), cpy = A(n), cqx = A(n), cqy = A(n);
    REP(i, d) cpx[mpa1[i]] = mpb1[c1[i]], cpy[mpa1[i]]=0;
    REP(i, n - d) cpx[mpa2[i]] = mpb2[c2[i]], cpy[mpa2[i]]=1;
    REP(i, n) r[i] = cpx[i];
    REP(i, n) cqx[cpx[i]] = i, cqy[cpx[i]] = cpy[i];
    int hi = n, lo = n, his = 0, los = 0;
    REP(i, n) {
      if(cqy[i] ^ (cqx[i] >= hi)) his--;
      while(hi > 0 && his < 0) {
        hi--;
        if(cpy[hi] ^ (cpx[hi] > i)) his++;
      }
      while(lo > 0 && los <= 0) {
        lo--;
        if(cpy[lo] ^ (cpx[lo] >= i)) los++;
      }
      if(los > 0 && hi == lo) r[lo] = i;
      if(cqy[i] ^ (cqx[i] >= lo)) los--;
    }
    ps = lps;
  }
  void subunit_monge_mult(I a, I b, I c, int n) {
    int lps = ps;
    I za = A(n), zb = A(n), res = A(n), vis = A(n), mpa = A(n), mpb = A(n), rb = A(n);
    MEM(vis, 0, n), MEM(mpa, -1, n), MEM(mpb, -1, n), MEM(rb, -1, n);
    int ca = n;
    IREP(i, n) if(a[i] != -1) vis[a[i]] = 1, za[--ca] = a[i], mpa[ca] = i;
    IREP(i, n) if(!vis[i]) za[--ca] = i;
    MEM(vis, -1, n);
    REP(i, n) if(b[i] != -1) vis[b[i]] = i;
    ca = 0;
    REP(i, n) if(vis[i] != -1) mpb[ca] = i, rb[vis[i]] = ca++;
    REP(i, n) if(rb[i] == -1) rb[i] = ca++;
    REP(i, n) zb[rb[i]] = i;
    unit_monge_mult(za, zb, res, n);
    MEM(c, -1, n);
    REP(i, n) if(mpa[i] != -1 && mpb[res[i]] != -1) c[mpa[i]] = mpb[res[i]];
    ps = lps;
  }
  void solve(I p, I ret, int n) {
    if(n == 1) return ret[0] = -1, void();
    int lps = ps, d = n / 2;
    I pl = A(d), pr = A(n - d);
    REP(i, d) pl[i] = p[i];
    REP(i, n - d) pr[i] = p[i + d];
    I vis = A(n); MEM(vis, -1, n);
    REP(i, d) vis[pl[i]] = i;
    I tl = A(d), tr = A(n - d), mpl = A(d), mpr = A(n - d);
    int ca = 0;
    REP(i, n) if(vis[i] != -1) mpl[ca] = i, tl[vis[i]] = ca++;
    ca = 0; MEM(vis, -1, n);
    REP(i, n - d) vis[pr[i]] = i;
    REP(i, n) if(vis[i] != -1) mpr[ca] = i, tr[vis[i]] = ca++;
    I vl = A(d), vr = A(n - d);
    solve(tl, vl, d), solve(tr, vr, n - d);
    I sl = A(n), sr = A(n);
    iota(sl, sl + n, 0); iota(sr, sr + n, 0);
    REP(i, d) sl[mpl[i]] = (vl[i] == -1 ? -1 : mpl[vl[i]]);
    REP(i, n - d) sr[mpr[i]] = (vr[i] == -1 ? -1 : mpr[vr[i]]);
    subunit_monge_mult(sl, sr, ret, n);
    ps = lps;
  }
  vi solve() {
    solve(invp, res_monge, n);
    vi fenw(n + 1);
    IREP(i, n) {
      if(res_monge[i] != -1) {
        for(int p = res_monge[i] + 1; p <= n; p += p & -p) fenw[p]++;
      }
      for(auto& z : qry[i]){
        auto [id, c] = z;
        for(int p = id; p; p -= p & -p) ans[c] -= fenw[p];
      }
    }
    free(pool);
    return ans;
  }
};
```

2.15 treap

```
struct Node {
  bool rev = false;
  int sz = 1, pri = rng();
  Node *l = NULL, *r = NULL, *p = NULL;
  // TODO
}
void pull(Node*& v) {
  v->sz = 1 + size(v->l) + size(v->r);
  // TODO
}
void push(Node*& v) {
  if(v->rev) {
    swap(v->l, v->r);
    if(v->l) v->l->rev ^= 1;
    if(v->r) v->r->rev ^= 1;
    v->rev = false;
  }
}
Node* merge(Node* a, Node* b) {
  if(!a || !b) return (a ? a : b);
  push(a), push(b);
  if(a->pri > b->pri) {
    a->r = merge(a->r, b);
    pull(a); return a;
  } else {
    b->l = merge(a, b->l);
    pull(b); return b;
  }
}
pair<Node*, Node*> split(Node* v, int k) {
  if(!v) return {NULL, NULL};
  push(v);
  if(size(v->l) >= k) {
    auto p = split(v->l, k);
    if(p.first) p.first->p = NULL;
    v->l = p.second;
    pull(v); return {p.first, v};
  } else {
    auto p = split(v->r, k - size(v->l) - 1);
    if(p.second) p.second->p = NULL;
    v->r = p.first;
    pull(v); return {v, p.second};
  }
}
int get_position(Node* v) { // 0-indexed
  int k = (v->l != NULL ? v->l->sz : 0);
  while(v->p != NULL) {
    if(v == v->p->r) {
      k++;
      if(v->p->l != NULL) k += v->p->l->sz;
```

```
    }
    v = v->p;
  }
  return k;
}
```

2.16 union-of-rectangles

```
// 2
// 1 10 1 10
// 0 2 0 2
// ans = 84
vector<int> vx, vy;
struct q { int piv, s, e, x; };
struct tree {
  vector<int> seg, tag;
  tree(int _n) : seg(_n * 16), tag(_n * 16) {}
  void add(int ql, int qr, int x, int v, int l, int r) {
    if(qr <= l || r <= ql) return;
    if(ql <= l && r <= qr) {
      tag[v] += x;
      if(tag[v] == 0) {
        if(l != r) seg[v] = seg[2 * v] + seg[2 * v + 1];
        else seg[v] = 0;
      } else seg[v] = vx[r] - vx[l];
    } else {
      int mid = (l + r) / 2;
      add(ql, qr, x, 2 * v, l, mid);
      add(ql, qr, x, 2 * v + 1, mid, r);
      if(tag[v] == 0 && l != r) seg[v] = seg[2 * v] + seg[2 * v + 1];
    }
  }
  int q() { return seg[1]; }
};
int main() {
  int n; cin >> n;
  vector<int> x1(n), x2(n), y_(n), y2(n);
  for (int i = 0; i < n; i++) {
    cin >> x1[i] >> x2[i] >> y_[i] >> y2[i]; // L R D U
    vx.pb(x1[i]), vx.pb(x2[i]);
    vy.pb(y_[i]), vy.pb(y2[i]);
  }
  vx = sort_unique(vx);
  vy = sort_unique(vy);
  vector<q> a(2 * n);
  REP(i, n) {
    x1[i] = lower_bound(ALL(vx), x1[i]) - vx.begin();
    x2[i] = lower_bound(ALL(vx), x2[i]) - vx.begin();
    y_[i] = lower_bound(ALL(vy), y_[i]) - vy.begin();
    y2[i] = lower_bound(ALL(vy), y2[i]) - vy.begin();
    a[2 * i] = {y_[i], x1[i], x2[i], +1};
    a[2 * i + 1] = {y2[i], x1[i], x2[i], -1};
  }
  sort(ALL(a), [](q a, q b) { return a.piv < b.piv; });
  tree seg(n);
  ll ans = 0;
  REP(i, 2 * n) {
    int j = i;
    while(j < 2 * n && a[i].piv == a[j].piv) {
      seg.add(a[j].s, a[j].e, a[j].x, 1, 0, vx.size());
      j++;
    }
    if(a[i].piv + 1 != SZ(vy)) ans += 1LL * seg.q() * (vy[a[i].piv + 1] - vy[a[i].piv]);
    i = j - 1;
  }
  cout << ans << "\n";
}
```

2.17 wavelet-tree

```
template<class T>
struct wavelet_tree {
  int n, log;
  vector<T> vals;
  vi sums;
  vector<ull> bits;
  inline void set_bit(int i, ull v) { bits[i >> 6] |= (v << (i & 63)); }
  inline int get_sum(int i) const { return sums[i >> 6] + __builtin_popcountll(bits[i >> 6] & ((1ULL << (i & 63)) - 1)); }
  wavelet_tree(const vector<T>& _v) : n(SZ(_v)) {
    vals = sort_unique(_v);
    log = __lg(2 * vals.size() - 1);
    bits.resize((log * n + 64) >> 6, 0ULL);
    sums.resize(SZ(bits), 0);
    vi v(SZ(_v)), cnt(SZ(vals) + 1);
    REP(i, SZ(v)) {
      v[i] = lower_bound(ALL(vals), _v[i]) - vals.begin();
      cnt[v[i] + 1] += 1;
    }
    partial_sum(ALL(cnt) - 1, cnt.begin());
    REP(j, log) {
      for(int i : v) {
        int tmp = i >> (log - 1 - j);
        int pos = (tmp >> 1) << (log - j);
        set_bit(j * n + cnt[pos], tmp & 1);
        cnt[pos]++;
      }
      for(int i : v) cnt[(i >> (log - j)) << (log - j)]--;
    }
    for(int i = 1; i < (int) sums.size(); i++) sums[i] = sums[i - 1] + __builtin_popcountll(bits[i - 1]);
  }

  T get_kth(int a, int b, int k) {
    for(int j = 0, ia = 0, ib = n, res = 0;; j++) {
      if(j == log) return vals[res];
      int cnt_ia = get_sum(n * j + ia);
      int cnt_a = get_sum(n * j + a);
      int cnt_b = get_sum(n * j + b);
      int cnt_ib = get_sum(n * j + ib);
      int ab_zeros = (b - a) - (cnt_b - cnt_a);
      if(ab_zeros > k) {
        res <<= 1;
        ib -= cnt_ib - cnt_ia;
        a -= cnt_a - cnt_ia;
        b -= cnt_b - cnt_ia;
      } else {
        res = (res << 1) | 1;
        k -= ab_zeros;
        ia += (ib - ia) - (cnt_ib - cnt_ia);
        a += (ib - a) - (cnt_ib - cnt_a);
        b += (ib - b) - (cnt_ib - cnt_b);
      }
    }
  }
};
```

3 Flow-Matching

3.1 bipartite-matching

```
struct bipartite_matching {
  int n, m; // 二分圖左右人數 (0 ~ n-1), (0 ~ m-1)
  vector<vi> g;
  vi lhs, rhs, dist; // i 與 lhs[i] 配對 (lhs[i] == -1 代表沒有配對)
  bipartite_matching(int _n, int _m) : n(_n), m(_m), g(_n), lhs(_n, -1), rhs(_m, -1), dist(_n) {}
  void add_edge(int u, int v) { g[u].pb(v); }
  void bfs() {
    queue<int> q;
    REP(i, n) {
      if(lhs[i] == -1) {
        q.push(i);
        dist[i] = 0;
      } else {
        dist[i] = -1;
      }
    }
    while(!q.empty()) {
      int u = q.front(); q.pop();
      for(auto v : g[u]) {
        if(rhs[v] != -1 && dist[rhs[v]] == -1) {
          dist[rhs[v]] = dist[u] + 1;
          q.push(rhs[v]);
        }
      }
    }
  }
  bool dfs(int u) {
    for(auto v : g[u]) {
      if(rhs[v] == -1) {
        rhs[lhs[u] = v] = u;
        return true;
      }
    }
    for(auto v : g[u]) {
      if(dist[rhs[v]] == dist[u] + 1 && dfs(rhs[v])) {
        rhs[lhs[u] = v] = u;
        return true;
      }
    }
    return false;
  }
  int solve() {
    int ans = 0;
    while(true) {
      bfs();
      int aug = 0;
      REP(i, n) if(lhs[i] == -1) aug += dfs(i);
      if(!aug) break;
      ans += aug;
    }
    return ans;
  }
};
```

3.2 Dinic-LowerBound

```
template<class T>
struct DinicLowerBound {
  using Maxflow = Dinic<T>;
  int n;
  Maxflow d;
  vector<T> in;
  DinicLowerBound(int _n) : n(_n), d(_n + 2), in(_n) {}
  int add_edge(int from, int to, T low, T high) {
    assert(0 <= low && low <= high);
    in[from] -= low, in[to] += low;
    return d.add_edge(from, to, high - low);
  }
  T flow(int s, int t) {
    T sum = 0;
    REP(i, n) {
      if(in[i] > 0) {
        d.add_edge(n, i, in[i]);
        sum += in[i];
      }
      if(in[i] < 0) d.add_edge(i, n + 1, -in[i]);
    }
    d.add_edge(t, s, numeric_limits<T>::max());
    if(d.flow(n, n + 1) < sum) return -1;
    return d.flow(s, t);
  }
};
```

3.3 Dinic

```cpp
template<class T>
class Dinic {
public:
  struct Edge {
    int from, to;
    T cap;
    Edge(int x, int y, T z) : from(x), to(y), cap(z) {}
  };
  constexpr T INF = 1e9;
  int n;
  vector<Edge> edges;
  vector<vi> g;
  vi cur, h; // h : level graph
  Dinic(int _n) : n(_n), g(_n) {}
  void add_edge(int u, int v, T c) {
    g[u].pb(sz(edges));
    edges.eb(u, v, c);
    g[v].pb(sz(edges));
    edges.eb(v, u, 0);
  }
  bool bfs(int s, int t) {
    h.assign(n, -1);
    queue<int> q;
    h[s] = 0;
    q.push(s);
    while(!q.empty()) {
      int u = q.front(); q.pop();
      for(int i : g[u]) {
        const auto& e = edges[i];
        int v = e.to;
        if(e.cap > 0 && h[v] == -1) {
          h[v] = h[u] + 1;
          if(v == t) return true;
          q.push(v);
        }
      }
    }
    return false;
  }
  T dfs(int u, int t, T f) {
    if(u == t) return f;
    T r = f;
    for(int& i = cur[u]; i < sz(g[u]); ++i) {
      int j = g[u][i];
      const auto& e = edges[j];
      int v = e.to;
      T c = e.cap;
      if(c > 0 && h[v] == h[u] + 1) {
        T a = dfs(v, t, min(r, c));
        edges[j].cap -= a;
        edges[j ^ 1].cap += a;
        if((r -= a) == 0) return f;
      }
    }
    return f - r;
  }
  T flow(int s, int t, T f = INF) {
    T ans = 0;
    while(f > 0 && bfs(s, t)) {
      cur.assign(n, 0);
      T cur = dfs(s, t, f);
      ans += cur;
      f -= cur;
    }
    return ans;
  }
};
```

3.4 Flow 建模

- Maximum/Minimum flow with lower bound / Circulation problem
  1. Construct super source S and sink T.
  2. For each edge (x, y, l, u), connect $x \to y$ with capacity $u - l$.
  3. For each vertex v, denote by $in(v)$ the difference between the sum of incoming lower bounds and the sum of outgoing lower bounds.
  4. If $in(v) > 0$, connect $S \to v$ with capacity $in(v)$, otherwise, connect $v \to T$ with capacity $-in(v)$.
     - To maximize, connect $t \to s$ with capacity ∞ (skip this in circulation problem), and let f be the maximum flow from S to T. If $f \neq \sum_{v \in V, in(v)>0} in(v)$, there's no solution. Otherwise, the maximum flow from s to t is the answer.
     - To minimize, let f be the maximum flow from S to T. Connect $t \to s$ with capacity ∞ and let the flow from S to T be f'. If $f + f' \neq \sum_{v \in V, in(v)>0} in(v)$, there's no solution. Otherwise, f' is the answer.
  5. The solution of each edge e is $l_e + f_e$, where f_e corresponds to the flow of edge e on the graph.
- Construct minimum vertex cover from maximum matching M on bipartite graph (X, Y)
  1. Redirect every edge: $y \to x$ if $(x, y) \in M$, $x \to y$ otherwise.
  2. DFS from unmatched vertices in X.
  3. $x \in X$ is chosen iff x is unvisited.
  4. $y \in Y$ is chosen iff y is visited.
- Minimum cost cyclic flow
  1. Consruct super source S and sink T
  2. For each edge (x, y, c), connect $x \to y$ with $(cost, cap) = (c, 1)$ if $c > 0$, otherwise connect $y \to x$ with $(cost, cap) = (-c, 1)$
  3. For each edge with $c < 0$, sum these cost as K, then increase $d(y)$ by 1, decrease $d(x)$ by 1
  4. For each vertex v with $d(v) > 0$, connect $S \to v$ with $(cost, cap) = (0, d(v))$
  5. For each vertex v with $d(v) < 0$, connect $v \to T$ with $(cost, cap) = (0, -d(v))$
  6. Flow from S to T, the answer is the cost of the flow $C + K$
- Maximum density induced subgraph
  1. Binary search on answer, suppose we're checking answer T
  2. Construct a max flow model, let K be the sum of all weights
  3. Connect source $s \to v, v \in G$ with capacity K
  4. For each edge (u, v, w) in G, connect $u \to v$ and $v \to u$ with capacity w
  5. For $v \in G$, connect it with sink $v \to t$ with capacity $K + 2T - (\sum_{e \in E(v)} w(e)) - 2w(v)$
  6. T is a valid answer if the maximum flow $f < K|V|$
- Minimum weight edge cover
  1. For each $v \in V$ create a copy v', and connect $u' \to v'$ with weight $w(u, v)$.
  2. Connect $v \to v'$ with weight $2\mu(v)$, where $\mu(v)$ is the cost of the cheapest edge incident to v.
  3. Find the minimum weight perfect matching on G'.
- Project selection problem
  1. If $p_v > 0$, create edge (s, v) with capacity p_v; otherwise, create edge (v, t) with capacity $-p_v$.
  2. Create edge (u, v) with capacity w with w being the cost of choosing u without choosing v.
  3. The mincut is equivalent to the maximum profit of a subset of projects.
- 0/1 quadratic programming

$$\sum_x c_x x + \sum_y c_y \bar{y} + \sum_{xy} c_{xy} x \bar{y} + \sum_{xyx'y'} c_{xyx'y'} (x\bar{y} + x'\bar{y'})$$

  can be minimized by the mincut of the following graph:
  1. Create edge (x, t) with capacity c_x and create edge (s, y) with capacity c_y.
  2. Create edge (x, y) with capacity c_{xy}.
  3. Create edge (x, y) and edge (x', y') with capacity $c_{xyx'y'}$.

3.5 general-matching

```cpp
struct GeneralMaxMatch {
  int n;
  vector<pii> es;
  vi g, vis, mate; // i 與 mate[i] 配對 (mate[i] == -1 代表沒有匹配)
  GeneralMaxMatch(int n) : n(n), g(n, -1), mate(n, -1) {}
  bool dfs(int u) {
    if(vis[u]) return false;
    vis[u] = true;
    for(int ei = g[u]; ei != -1; ) {
      auto [x, y] = es[ei]; ei = y;
      if(mate[x] == -1) {
        mate[mate[u] = x] = u;
        return true;
      }
    }
    for(int ei = g[u]; ei != -1; ) {
      auto [x, y] = es[ei]; ei = y;
      int nu = mate[x];
      mate[mate[u] = x] = u;
      mate[nu] = -1;
      if(dfs(nu)) return true;
      mate[mate[nu] = x] = nu;
      mate[u] = -1;
    }
    return false;
  }
  void add_edge(int a, int b) {
    auto f = [&](int a, int b) {
      es.eb(b, g[a]);
      g[a] = sz(es) - 1;
    };
    f(a, b); f(b, a);
  }
  int solve() {
    vi o(n);
    iota(all(o), 0);
    int ans = 0;
    REP(it, 100) {
      shuffle(all(o), rng);
      vis.assign(n, false);
      for(auto i : o) if(mate[i] == -1) ans += dfs(i);
    }
    return ans;
  }
};
```

3.6 general-weighted-max-matching

```cpp
// 1-based QQ
struct WeightGraph {
  static const int inf = INT_MAX;
  static const int maxn = 514;
  struct edge {
    int u, v, w;
    edge() {}
    edge(int u, int v, int w): u(u), v(v), w(w) {}
  };
  int n, n_x;
  edge g[maxn * 2][maxn * 2];
  int lab[maxn * 2];
  int match[maxn * 2], slack[maxn * 2], st[maxn * 2], pa[maxn * 2];
  int flo_from[maxn * 2][maxn + 1], S[maxn * 2], vis[maxn * 2];
  vector<int> flo[maxn * 2];
  queue<int> q;
  int e_delta(const edge &e) { return lab[e.u] + lab[e.v] - g[e.u][e.v].w * 2; }
  void update_slack(int u, int x) { if(!slack[x] || e_delta(g[u][x]) < e_delta(g[slack[x]][x])) slack[x] = u; }
  void set_slack(int x) {
    slack[x] = 0;
    REP(u, n) if(g[u + 1][x].w > 0 && st[u + 1] != x && S[st[u + 1]] == 0) update_slack(u + 1, x);
  }
  void q_push(int x) {
    if(x <= n) q.push(x);
    else REP(i, SZ(flo[x])) q_push(flo[x][i]);
  }
  void set_st(int x, int b) {
    st[x] = b;
```

```cpp
    if(x > n) REP(i, SZ(flo[x])) set_st(flo[x][i], b);
  }
  int get_pr(int b, int xr) {
    int pr = find(ALL(flo[b]), xr) - flo[b].begin();
    if(pr % 2 == 1) {
      reverse(1 + ALL(flo[b]));
      return SZ(flo[b]) - pr;
    }
    return pr;
  }
  void set_match(int u, int v) {
    match[u] = g[u][v].v;
    if(u <= n) return;
    edge e = g[u][v];
    int xr = flo_from[u][e.u], pr = get_pr(u, xr);
    for(int i = 0; i < pr; ++i) set_match(flo[u][i], flo[u][i ^ 1]);
    set_match(xr, v);
    rotate(flo[u].begin(), flo[u].begin() + pr, flo[u].end());
  }
  void augment(int u, int v) {
    while(true) {
      int xnv = st[match[u]];
      set_match(u, v);
      if(!xnv) return;
      set_match(xnv, st[pa[xnv]]);
      u = st[pa[xnv]], v = xnv;
    }
  }
  int get_lca(int u, int v) {
    static int t = 0;
    for(++t; u || v; swap(u, v)) {
      if(u == 0) continue;
      if(vis[u] == t) return u;
      vis[u] = t;
      if(u = st[match[u]]) u = st[pa[u]];
    }
    return 0;
  }
  void add_blossom(int u, int lca, int v) {
    int b = n + 1;
    while(b <= n_x && st[b]) ++b;
    if(b > n_x) n_x++;
    lab[b] = S[b] = 0;
    match[b] = match[lca];
    flo[b].clear(); flo[b].pb(lca);
    for(int x = u, y; x != lca; x = st[pa[y]]) flo[b].pb(x), flo[b].pb(y = st[match[x]]), q_push(y);
    reverse(1 + ALL(flo[b]));
    for(int x = v, y; x != lca; x = st[pa[y]]) flo[b].pb(x), flo[b].pb(y = st[match[x]]), q_push(y);
    set_st(b, b);
    REP(x, n_x) g[b][x + 1].w = g[x + 1][b].w = 0;
    REP(x, n) flo_from[b][x + 1] = 0;
    REP(i, SZ(flo[b])) {
      int xs = flo[b][i];
      REP(x, n_x) if(g[b][x + 1].w == 0 || e_delta(g[xs][x + 1]) < e_delta(g[b][x + 1])) g[b][x + 1] = g[xs][x + 1], g[x + 1][b] = g[x + 1][xs];
      REP(x, n) if(flo_from[xs][x + 1]) flo_from[b][x + 1] = xs;
    }
    set_slack(b);
  }
  void expand_blossom(int b) {
    REP(i, SZ(flo[b])) set_st(flo[b][i], flo[b][i]);
    int xr = flo_from[b][g[b][pa[b]].u], pr = get_pr(b, xr);
    for(int i = 0; i < pr; i += 2) {
      int xs = flo[b][i], xns = flo[b][i + 1];
      pa[xs] = g[xns][xs].u;
      S[xs] = 1, S[xns] = 0;
      slack[xs] = 0, set_slack(xns);
      q_push(xns);
    }
    S[xr] = 1, pa[xr] = pa[b];
    for(size_t i = pr + 1; i < SZ(flo[b]); ++i) {
      int xs = flo[b][i];
      S[xs] = -1, set_slack(xs);
    }
    st[b] = 0;
  }
  bool on_found_edge(const edge &e) {
    int u = st[e.u], v = st[e.v];
    if(S[v] == -1) {
      pa[v] = e.u, S[v] = 1;
      int nu = st[match[v]];
      slack[v] = slack[nu] = 0;
      S[nu] = 0, q_push(nu);
    } else if(S[v] == 0) {
      int lca = get_lca(u, v);
      if(!lca) return augment(u,v), augment(v,u), true;
      else add_blossom(u, lca, v);
    }
    return false;
  }
  bool matching() {
    memset(S + 1, -1, sizeof(int) * n_x);
    memset(slack + 1, 0, sizeof(int) * n_x);
    q = queue<int>();
    REP(x, n_x) if(st[x + 1] == x + 1 && !match[x + 1]) pa[x + 1] = 0, S[x + 1] = 0, q_push(x + 1);
    if(q.empty()) return false;
    while(true) {
      while(!q.empty()) {
        int u = q.front(); q.pop();
        if(S[st[u]] == 1) continue;
        for(int v = 1; v <= n; ++v)
          if(g[u][v].w > 0 && st[u] != st[v]) {
            if(e_delta(g[u][v]) == 0) {
              if(on_found_edge(g[u][v])) return true;
            } else update_slack(u, st[v]);
          }
      }
      int d = inf;
      for(int b = n + 1; b <= n_x; ++b) if(st[b] == b && S[b] == 1) d = min(d, lab[b] / 2);
      for(int x = 1; x <= n_x; ++x) {
        if(st[x] == x && slack[x]) {
          if(S[x] == -1) d = min(d, e_delta(g[slack[x]][x]));
          else if(S[x] == 0) d = min(d, e_delta(g[slack[x]][x]) / 2);
        }
      }
      REP(u, n) {
        if(S[st[u + 1]] == 0) {
          if(lab[u + 1] <= d) return 0;
          lab[u + 1] -= d;
        } else if(S[st[u + 1]] == 1) lab[u + 1] += d;
      }
      for(int b = n + 1; b <= n_x; ++b)
        if(st[b] == b) {
          if(S[st[b]] == 0) lab[b] += d * 2;
          else if(S[st[b]] == 1) lab[b] -= d * 2;
        }
      q = queue<int>();
      for(int x = 1; x <= n_x; ++x)
        if(st[x] == x && slack[x] && st[slack[x]] != x && e_delta(g[slack[x]][x]) == 0)
          if(on_found_edge(g[slack[x]][x])) return true;
      for(int b = n + 1; b <= n_x; ++b)
        if(st[b] == b && S[b] == 1 && lab[b] == 0) expand_blossom(b);
    }
    return false;
  }
  pair<ll, int> solve() {
    memset(match + 1, 0, sizeof(int) * n);
    n_x = n;
    int n_matches = 0;
    ll tot_weight = 0;
    for(int u = 0; u <= n; ++u) st[u] = u, flo[u].clear();
    int w_max = 0;
    for(int u = 1; u <= n; ++u)
      for(int v = 1; v <= n; ++v) {
        flo_from[u][v] = (u == v ? u : 0);
        w_max = max(w_max, g[u][v].w);
      }
    for(int u = 1; u <= n; ++u) lab[u] = w_max;
    while(matching()) ++n_matches;
    for(int u = 1; u <= n; ++u)
      if(match[u] && match[u] < u)
        tot_weight += g[u][match[u]].w;
    return make_pair(tot_weight, n_matches);
  }
  void add_edge(int u, int v, int w) { g[u][v].w = g[v][u].w = w; }
  void init(int _n) : n(_n) {
    REP(u, n) REP(v, n) g[u + 1][v + 1] = edge(u + 1, v + 1, 0);
  }
};
```

3.7 KM

```cpp
template<class T>
struct KM {
  static constexpr T INF = numeric_limits<T>::max();
  int n, ql, qr;
  vector<vector<T>> w;
  vector<T> hl, hr, slk;
  vi fl, fr, pre, qu;
  vector<bool> vl, vr;
  KM(int n) : n(n), w(n, vector<T>(n, -INF)), hl(n), hr(n), slk(n), fl(n), fr(n), pre(n), qu(n), vl(n), vr(n) {}
  void add_edge(int u, int v, int x) { w[u][v] = x; } // 最小值要加負號
  bool check(int x) {
    vl[x] = 1;
    if(fl[x] != -1) return vr[qu[qr++] = fl[x]] = 1;
    while(x != -1) swap(x, fr[fl[x] = pre[x]]);
    return 0;
  }
  void bfs(int s) {
    fill(all(slk), INF);
    fill(all(vl), 0);
    fill(all(vr), 0);
    ql = qr = 0, qu[qr++] = s, vr[s] = 1;
    while(true) {
      T d;
      while(ql < qr) {
        for(int x = 0, y = qu[ql++]; x < n; ++x) {
          if(!vl[x] && slk[x] >= (d = hl[x] + hr[y] - w[x][y])) {
            pre[x] = y;
            if(d) slk[x] = d;
            else if(!check(x)) return;
          }
        }
      }
      d = INF;
      REP(x, n) if(!vl[x] && d > slk[x]) d = slk[x];
      REP(x, n) {
        if(vl[x]) hl[x] += d;
        else slk[x] -= d;
        if(vr[x]) hr[x] -= d;
      }
      REP(x, n) if(!vl[x] && !slk[x] && !check(x)) return;
    }
  }
  T solve() {
    fill(all(fl), -1);
    fill(all(fr), -1);
    fill(all(hr), 0);
    REP(i, n) hl[i] = *max_element(all(w[i]));
    REP(i, n) bfs(i);
    T ans = 0;
    REP(i, n) ans += w[i][fl[i]]; // i 跟 fl[i] 配對
    return ans;
```

```
  }
};
```

3.8 max-clique

```
template<int V>
struct max_clique {
  using B = bitset<V>;
  int n = 0;
  vector<B> g, buf;
  struct P {
    int idx, col, deg;
    P(int a, int b, int b) : idx(a), col(b), deg(c) {}
  };
  max_clique(int _n) : n(_n), g(_n), buf(_n) {}
  void add_edge(int a, int b) {
    assert(a != b);
    g[a][b] = g[b][a] = 1;
  }
  vector<int> now, clique;
  void dfs(vector<P>& rem){
    if(SZ(clique) < SZ(now)) clique = now;
    sort(ALL(rem), [](P a, P b) { return a.deg > b.deg; });
    int max_c = 1;
    for(auto& p : rem){
      p.col = 0;
      while((g[p.idx] & buf[p.col]).any()) p.col++;
      max_c = max(max_c, p.idx + 1);
      buf[p.col][p.idx] = 1;
    }
    REP(i, max_c) buf[i].reset();
    sort(ALL(rem), [&](P a, P b) { return a.col < b.col; });
    for(; !rem.empty(); rem.pop_back()){
      auto& p = rem.back();
      if(now.size() + p.col + 1 <= clique.size()) break;
      vector<P> nrem;
      B bs;
      for(auto& q : rem){
        if(g[p.idx][q.idx]){
          nrem.emplace_back(q.idx, -1, 0);
          bs[q.idx] = 1;
        }
      }
      for(auto& q : nrem) q.deg = (bs & g[q.idx]).count();
      now.emplace_back(p.idx);
      dfs(nrem);
      now.pop_back();
    }
  }
  vector<int> solve(){
    vector<P> remark;
    REP(i, n) remark.emplace_back(i, -1, SZ(g[i]));
    dfs(remark);
    return clique;
  }
};
```

3.9 MCMF

```
template<class S, class T>
class MCMF {
public:
  struct Edge {
    int from, to;
    S cap;
    T cost;
    Edge(int u, int v, S x, T y) : from(u), to(v), cap(x), cost(y) {}
  };
  const ll INF = 1e18L;
  int n;
  vector<Edge> edges;
  vector<vi> g;
  vector<T> d;
  vector<bool> inq;
  vi pedge;
  MCMF(int _n) : n(_n), g(_n), d(_n), inq(_n), pedge(_n) {}
  void add_edge(int u, int v, S cap, T cost) {
    g[u].pb(sz(edges));
    edges.eb(u, v, cap, cost);
    g[v].pb(sz(edges));
    edges.eb(v, u, 0, -cost);
  }
  bool spfa(int s, int t) {
    bool found = false;
    fill(all(d), INF);
    d[s] = 0;
    inq[s] = true;
    queue<int> q;
    q.push(s);
    while(!q.empty()) {
      int u = q.front(); q.pop();
      if(u == t) found = true;
      inq[u] = false;
      for(auto& id : g[u]) {
        const auto& e = edges[id];
        if(e.cap > 0 && d[u] + e.cost < d[e.to]) {
          d[e.to] = d[u] + e.cost;
          pedge[e.to] = id;
          if(!inq[e.to]) {
            q.push(e.to);
            inq[e.to] = true;
          }
        }
      }
    }
    return found;
  }
  pair<S, T> flow(int s, int t, S f = INF) {
    S cap = 0;
    T cost = 0;
    while(f > 0 && spfa(s, t)) {
      S send = f;
      int u = t;
      while(u != s) {
        const Edge& e = edges[pedge[u]];
        send = min(send, e.cap);
        u = e.from;
      }
      u = t;
      while(u != s) {
        Edge& e = edges[pedge[u]];
        e.cap -= send;
        Edge& b = edges[pedge[u] ^ 1];
        b.cap += send;
        u = e.from;
      }
      cap += send;
      f -= send;
      cost += send * d[t];
    }
    return {cap, cost};
  }
};
```

3.10 minimum-general-weighted-perfect-matching

```
struct Graph {
  // Minimum General Weighted Matching (Perfect Match) 0-base
  static const int MXN = 105;
  int n, edge[MXN][MXN];
  int match[MXN],dis[MXN],onstk[MXN];
  vector<int> stk;
  void init(int _n) {
    n = _n;
    for(int i=0; i<n; i++)
      for(int j=0; j<n; j++)
        edge[i][j] = 0;
  }
  void add_edge(int u, int v, int w) { edge[u][v] = edge[v][u] = w; }
  bool SPFA(int u){
    if(onstk[u]) return true;
    stk.push_back(u);
    onstk[u] = 1;
    for(int v=0; v<n; v++){
      if(u != v && match[u] != v && !onstk[v]){
        int m = match[v];
        if(dis[m] > dis[u] - edge[v][m] + edge[u][v]){
          dis[m] = dis[u] - edge[v][m] + edge[u][v];
          onstk[v] = 1;
          stk.push_back(v);
          if(SPFA(m)) return true;
          stk.pop_back();
          onstk[v] = 0;
        }
      }
    }
    onstk[u] = 0;
    stk.pop_back();
    return false;
  }
  int solve() {
    for(int i = 0; i < n; i += 2) match[i] = i + 1, match[i+1] = i;
    while(true) {
      int found = 0;
      for(int i=0; i<n; i++) dis[i] = onstk[i] = 0;
      for(int i=0; i<n; i++){
        stk.clear();
        if(!onstk[i] && SPFA(i)){
          found = 1;
          while(stk.size()>=2){
            int u = stk.back(); stk.pop_back();
            int v = stk.back(); stk.pop_back();
            match[u] = v;
            match[v] = u;
          }
        }
      }
      if(!found) break;
    }
    int ans = 0;
    for(int i=0; i<n; i++) ans += edge[i][match[i]];
    return ans / 2;
  }
}graph;
```

4 Geometry

4.1 closest-pair

```
const ll INF = 9e18L + 5;
vector<P> a;
sort(all(a), [](P a, P b) { return a.x < b.x; });
ll SQ(ll x) { return x * x; }
ll solve(int l, int r) {
  if(l + 1 == r) return INF;
  int m = (l + r) / 2;
  ll midx = a[m].x;
  ll d = min(solve(l, m), solve(m, r));
  inplace_merge(a.begin() + l, a.begin() + m, a.begin() + r, [](P a, P b) {
    return a.y < b.y;
  });
  vector<P> p;
  for(int i = l; i < r; ++i) if(SQ(a[i].x - midx) < d) p.pb(a[i]);
  REP(i, sz(p)) {
    for(int j = i + 1; j < sz(p); ++j) {
      d = min(d, SQ(p[i].x - p[j].x) + SQ(p[i].y - p[j].y));
      if(SQ(p[i].y - p[j].y) > d) break;
    }
  }
  return d; // 距離平方
}
```

4.2 convex-hull

```
void convex_hull(vector<P>& dots) {
  sort(all(dots));
  vector<P> ans(1, dots[0]);
  for(int it = 0; it < 2; it++, reverse(all(dots))) {
    for(int i = 1, t = sz(ans); i < sz(dots); ans.pb(dots[i++])) {
      while(sz(ans) > t && ori(ans[sz(ans) - 2], ans.back(), dots[i]) < 0) {
        ans.ppb();
      }
    }
  }
  ans.ppb();
  swap(ans, dots);
}
```

4.3 point-in-convex-hull

```
int point_in_convex_hull(const vector<P>& a, P p) {
  // -1 ON, 0 OUT, +1 IN
  // 要先逆時針排序
  int n = sz(a);
  if(btw(a[0], a[1], p) || btw(a[0], a[n - 1], p)) return -1;
  int l = 0, r = n - 1;
  while(l <= r) {
    int m = (l + r) / 2;
    auto a1 = cross(a[m] - a[0], p - a[0]);
    auto a2 = cross(a[(m + 1) % n] - a[0], p - a[0]);
    if(a1 >= 0 && a2 <= 0) {
      auto res = cross(a[(m + 1) % n] - a[m], p - a[m]);
      return res > 0 ? 1 : (res >= 0 ? -1 : 0);
    }
    if(a1 < 0) r = m - 1;
    else l = m + 1;
  }
  return 0;
}
```

4.4 point

```
using P = pair<ll, ll>;
P operator+(P a, P b) { return P{a.X + b.X, a.Y + b.Y}; }
P operator-(P a, P b) { return P{a.X - b.X, a.Y - b.Y}; }
P operator*(P a, ll b) { return P{a.X * b, a.Y * b}; }
P operator/(P a, ll b) { return P{a.X / b, a.Y / b}; }
ll dot(P a, P b) { return a.X * b.X + a.Y * b.Y; }
ll cross(P a, P b) { return a.X * b.Y - a.Y * b.X; }
ll abs2(P a) { return dot(a, a); }
double abs(P a) { return sqrt(abs2(a)); }
int sign(ll x) { return x < 0 ? -1 : (x == 0 ? 0 : 1); }
int ori(P a, P b, P c) { return sign(cross(b - a, c - a)); }
bool collinear(P a, P b, P c) { return sign(cross(a - c, b - c)) == 0; }
bool btw(P a, P b, P c) {
  if(!collinear(a, b, c)) return 0;
  return sign(dot(a - c, b - c)) <= 0;
}
bool seg_intersect(P a, P b, P c, P d) {
  int a123 = ori(a, b, c);
  int a124 = ori(a, b, d);
  int a341 = ori(c, d, a);
  int a342 = ori(c, d, b);
  if(a123 == 0 && a124 == 0) {
    return btw(a, b, c) || btw(a, b, d) || btw(c, d, a) || btw(c, d, b);
  }
  return a123 * a124 <= 0 && a341 * a342 <= 0;
}

P intersect(P a, P b, P c, P d) {
  int a123 = cross(b - a, c - a);
  int a124 = cross(b - a, d - a);
  return (d * a123 - c * a124) / (a123 - a124);
}
struct line { P A, B; };
P vec(line L) { return L.B - L.A; }
P projection(P p, line L) { return L.A + vec(L) / abs(vec(L)) * dot(p - L.A, vec(L)) / abs(vec(L)); }
```

4.5 polar-angle-sort

```
bool cmp(P a, P b) {
  #define ng(k) (sign(k.Y) < 0 || (sign(k.Y) == 0 && sign(k.X) < 0))
  int A = ng(a), B = ng(b);
  if(A != B) return A < B;
  if(sign(cross(a, b)) == 0) return abs2(a) < abs2(b);
  return sign(cross(a, b)) > 0;
}
```

5 Graph

5.1 2-SAT

```
struct two_sat {
  int n; SCC g;
  vector<bool> ans;
  two_sat(int _n) : n(_n), g(_n * 2) {}
  void add_or(int u, bool x, int v, bool y) {
    g.add_edge(2 * u + !x, 2 * v + y);
    g.add_edge(2 * v + !y, 2 * u + x);
  }
  bool solve() {
    ans.resize(n);
    auto id = g.solve();
    REP(i, n) {
      if(id[2 * i] == id[2 * i + 1]) return false;
      ans[i] = (id[2 * i] < id[2 * i + 1]);
    }
    return true;
  }
};
```

5.2 centroid-tree

```
pair<int, vector<vi>> centroid_tree(const vector<vi>& g) {
  int n = sz(g);
  vi siz(n);
  vector<bool> vis(n);
  auto dfs_sz = [&](auto f, int u, int p) -> void {
    siz[u] = 1;
    for(auto v : g[u]) {
      if(v == p || vis[v]) continue;
      f(f, v, u);
      siz[u] += siz[v];
    }
  };
  auto find_cd = [&](auto f, int u, int p, int all) -> int {
    for(auto v : g[u]) {
      if(v == p || vis[v]) continue;
      if(siz[v] * 2 > all) return f(f, v, u, all);
    }
    return u;
  };
  vector<vi> h(n);
  auto build = [&](auto f, int u) -> int {
    dfs_sz(dfs_sz, u, -1);
    int cd = find_cd(find_cd, u, -1, siz[u]);
    vis[cd] = true;
    for(auto v : g[cd]) {
      if(vis[v]) continue;
      int child = f(f, v);
      h[cd].pb(child);
    }
    return cd;
  };
  int root = build(build, 0);
  return {root, h};
}
```

5.3 chromatic-number

```
// vi to(n);
// to[u] |= 1 << v;
// to[v] |= 1 << u;
int chromatic_number(vi g) {
  constexpr int MOD = 998244353;
  int n = SZ(g);
  vector<int> I(1 << n); I[0] = 1;
  for(int s = 1; s < (1 << n); s++) {
    int v = __builtin_ctz(s), t = s ^ (1 << v);
    I[s] = (I[t] + I[t & ~g[v]]) % MOD;
  }
  auto f = I;
  for(int k = 1; k <= n; k++) {
    int sum = 0;
    REP(s, 1 << n) {
      if((__builtin_popcount(s) ^ n) & 1) sum -= f[s];
      else sum += f[s];
      sum = ((sum % MOD) + MOD) % MOD;
      f[s] = 1LL * f[s] * I[s] % MOD;
    }
    if(sum != 0) return k;
  }
  return 48763;
}
```

5.4 HLD

```
struct HLD {
  int n;
  vector<vi> g;
  vi siz, par, depth, top, tour, fi, id;
  sparse_table<pii, min> st;
  HLD(int _n) : n(_n), g(_n), siz(_n), par(_n), depth(_n), top(_n), fi(_n), id(_n) {
    tour.reserve(n);
  }
  void add_edge(int u, int v) {
    g[u].push_back(v);
    g[v].push_back(u);
  }
  void build(int root = 0) {
    par[root] = -1;
    top[root] = root;
    vector<pii> euler_tour;
    euler_tour.reserve(2 * n - 1);
    dfs_sz(root);
    dfs_link(euler_tour, root);
    st = sparse_table<pii, min>(euler_tour);
  }
  int get_lca(int u, int v) {
    int L = fi[u], R = fi[v];
    if(L > R) swap(L, R);
    return st.prod(L, R).second;
  }
  bool is_anc(int u, int v) {
    return id[u] <= id[v] && id[v] < id[u] + siz[u];
```

```
  }
  bool on_path(int a, int b, int x) {
    return (is_ancestor(x, a) || is_ancestor(x, b)) && is_ancestor(get_lca(a, b), x);
  }
  int get_dist(int u, int v) {
    return depth[u] + depth[v] - 2 * depth[get_lca(u, v)];
  }
  int kth_anc(int u, int k) {
    if(depth[u] < k) return -1;
    int d = depth[u] - k;
    while(depth[top[u]] > d) u = par[top[u]];
    return tour[id[u] + d - depth[u]];
  }
  int kth_node_on_path(int a, int b, int k) {
    int z = get_lca(a, b);
    int fi = depth[a] - depth[z];
    int se = depth[b] - depth[z];
    if(k < 0 || k > fi + se) return -1;
    if(k < fi) return kth_anc(a, k);
    return kth_anc(b, fi + se - k);
  }
  vector<pii> get_path(int u, int v, bool include_lca = true) {
    if(u == v && !include_lca) return {};
    vector<pii> seg;
    while(top[u] != top[v]) {
      if(depth[top[u]] > depth[top[v]]) swap(u, v);
      seg.eb(id[top[v]], id[v]);
      v = par[top[v]];
    }
    if(depth[u] > depth[v]) swap(u, v); // u is lca
    if(u != v || include_lca) seg.eb(id[u] + !include_lca, id[v]);
    return seg;
  }
  void dfs_sz(int u) {
    if(par[u] != -1) g[u].erase(find(all(g[u]), par[u]));
    siz[u] = 1;
    for(auto& v : g[u]) {
      par[v] = u;
      depth[v] = depth[u] + 1;
      dfs_sz(v);
      siz[u] += siz[v];
      if(siz[v] > siz[g[u][0]]) swap(v, g[u][0]);
    }
  }
  void dfs_link(vector<pii>& euler_tour, int u) {
    fi[u] = sz(euler_tour);
    id[u] = sz(tour);
    euler_tour.eb(depth[u], u);
    tour.pb(u);
    for(auto v : g[u]) {
      top[v] = (v == g[u][0] ? top[u] : v);
      dfs_link(euler_tour, v);
      euler_tour.eb(depth[u], u);
    }
  }
};
```

5.5 lowlink

```
struct lowlink {
  int n, cnt = 0, tecc_cnt = 0, tvcc_cnt = 0;
  vector<vector<pii>> g;
  vector<pii> edges;
  vi roots, id, low, tecc_id, tvcc_id;
  vector<bool> is_bridge, is_cut, is_tree_edge;
  lowlink(int _n) : n(_n), g(_n), is_cut(_n, false), id(_n, -1), low(_n, -1) {}
  void add_edge(int u, int v) {
    g[u].eb(v, sz(edges));
    g[v].eb(u, sz(edges));
    edges.eb(u, v);
    is_bridge.pb(false);
    is_tree_edge.pb(false);
    tvcc_id.pb(-1);
  }
  void dfs(int u, int peid = -1) {
    static vi stk;
    static int rid;
    if(peid < 0) rid = cnt;
    if(peid == -1) roots.pb(u);
    id[u] = low[u] = cnt++;
    for(auto [v, eid] : g[u]) {
      if(eid == peid) continue;
      if(id[v] < id[u]) stk.pb(eid);
      if(id[v] >= 0) {
        low[u] = min(low[u], id[v]);
      } else {
        is_tree_edge[eid] = true;
        dfs(v, eid);
        low[u] = min(low[u], low[v]);
        if((id[u] == rid && id[v] != rid + 1) || (id[u] != rid && low[v] >= id[u])) {
          is_cut[u] = true;
        }
        if(low[v] >= id[u]) {
          while(true) {
            int e = stk.back();
            stk.pop_back();
            tvcc_id[e] = tvcc_cnt;
            if(e == eid) break;
          }
          tvcc_cnt++;
        }
      }
    }
  }
  void build() {
    REP(i, n) if(id[i] < 0) dfs(i);
    REP(i, sz(edges)) {
      auto [u, v] = edges[i];
      if(id[u] > id[v]) swap(u, v);
      is_bridge[i] = (id[u] < low[v]);
    }
  }
  vector<vi> two_ecc() { // 邊雙
    tecc_cnt = 0;
    tecc_id.assign(n, -1);
    vi stk;
    REP(i, n) {
      if(tecc_id[i] != -1) continue;
      tecc_id[i] = tecc_cnt;
      stk.pb(i);
      while(sz(stk)) {
        int u = stk.back(); stk.pop_back();
        for(auto [v, eid] : g[u]) {
          if(tecc_id[v] >= 0 || is_bridge[eid]) {
            continue;
          }
          tecc_id[v] = tecc_cnt;
          stk.pb(v);
        }
      }
      tecc_cnt++;
    }
    vector<vi> comp(tecc_cnt);
    REP(i, n) comp[tecc_id[i]].pb(i);
    return comp;
  }
  vector<vi> bcc_vertices() { // 點雙
    vector<vi> comp(tvcc_cnt);
    REP(i, sz(edges)) {
      comp[tvcc_id[i]].pb(edges[i].first);
      comp[tvcc_id[i]].pb(edges[i].second);
    }
    for(auto& v : comp) {
      sort(all(v));
      v.erase(unique(all(v)), v.end());
    }
    REP(i, n) if(g[i].empty()) comp.pb({i});
    return comp;
  }
  vector<vi> bcc_edges() {
    vector<vi> ret(tvcc_cnt);
    REP(i, sz(edges)) ret[tvcc_id[i]].pb(i);
    return ret;
  }
};
```

5.6 manhattan-mst

```
template<class T> // [w, u, v]
vector<tuple<T, int, int>> manhattan_mst(vector<T> xs, vector<T> ys) {
  const int n = SZ(xs);
  vi idx(n);
  iota(ALL(idx), 0);
  vector<tuple<T, int, int>> ret;
  REP(s, 2) {
    REP(t, 2) {
      auto cmp = [&](int i, int j) {
        return xs[i] + ys[i] < xs[j] + ys[j]; };
      sort(ALL(idx), cmp);
      map<T, int> sweep;
      for(int i : idx) {
        for(auto it = sweep.lower_bound(-ys[i]); it != sweep.end(); it = sweep.erase(it)) {
          int j = it->second;
          if(xs[i] - xs[j] < ys[i] - ys[j]) break;
          ret.eb(abs(xs[i] - xs[j]) + abs(ys[i] - ys[j]), i, j);
        }
        sweep[-ys[i]] = i;
      }
      swap(xs, ys);
    }
    for(auto &x : xs) x = -x;
  }
  sort(ALL(ret));
  return ret;
}
```

5.7 SCC

```
struct SCC {
  int n;
  vector<vi> g, h;
  SCC(int _n) : n(_n), g(_n), h(_n) {}
  void add_edge(int u, int v) {
    g[u].pb(v);
    h[v].pb(u);
  }
  vi solve() { // 回傳縮點的編號
    vi id(n), top;
    top.reserve(n);
    #define GO if(id[v] == 0) dfs1(v);
    function<void(int)> dfs1 = [&](int u) {
      id[u] = 1;
      for(auto v : g[u]) GO;
      top.pb(u);
    };
    REP(v, n) GO;
    fill(all(id), -1);
    function<void(int, int)> dfs2 = [&](int u, int x) {
      id[u] = x;
      for(auto v : h[u]) {
        if(id[v] == -1) {
          dfs2(v, x);
        }
      }
    };
    for(int i = n - 1, cnt = 0; i >= 0; --i) {
      int u = top[i];
      if(id[u] == -1) {
        dfs2(u, cnt);
        cnt += 1;
      }
    }
    return id;
  }
};
```

5.8 triangle-sum

```
// Three vertices a < b < cconnected by three edges {a, b}, {a, c}, {b, c}. Find xa * xb * xc over all triangles.
int triangle_sum(vector<array<int, 2>> edges, vi x) {
  int n = SZ(x);
  vi deg(n);
  vector<vector<int>> g(n);
  for(auto& [u, v] : edges) {
    if(u > v) swap(u, v);
    deg[u]++, deg[v]++;
  }
  REP(i, n) g[i].reserve(deg[i]);
  for(auto [u, v] : edges) {
    if(deg[u] > deg[v]) swap(u, v);
    g[u].pb(v);
  }
  vi val(n);
  __int128 ans = 0;
  REP(a, n) {
    for(auto b : g[a]) val[b] = x[b];
    for(auto b : g[a]) {
      ll tmp = 0;
      for(auto c : g[b]) tmp += val[c];
      ans += __int128(tmp) * x[a] * x[b];
    }
    for(auto b : g[a]) val[b] = 0;
  }
  return ans % mod;
}
```

6 Math

6.1 Aliens

```
template<class Func, bool MAX>
ll Aliens(ll l, ll r, int k, Func f) {
  while(l < r) {
    ll m = l + (r - l) / 2;
    auto [score, op] = f(m);
    if(op == k) return score + m * k * (MAX ? +1 : -1);
    if(op < k) r = m;
    else l = m + 1;
  }
  return f(l).first + l * k * (MAX ? +1 : -1);
}
```

6.2 Berlekamp-Massey

```
// - [1, 2, 4, 8, 16] -> (1, [1, -2])
// - [1, 1, 2, 3, 5, 8] -> (2, [1, -1, -1])
// - [0, 0, 0, 0, 1] -> (5, [1, 0, 0, 0, 0, 998244352]) (mod 998244353)
// - [] -> (0, [1])
// - [0, 0, 0] -> (0, [1])
// - [-2] -> (1, [1, 2])
template<class T>
pair<int, vector<T>> BM(const vector<T>& S) {
  using poly = vector<T>;
  int N = SZ(S);
  poly C_rev{1}, B{1};
  int L = 0, m = 1;
  T b = 1;
  auto adjust = [](poly C, const poly &B, T d, T b, int m) -> poly {
    C.resize(max(SZ(C), SZ(B) + m));
    T a = d / b;
    REP(i, SZ(B)) C[i + m] -= a * B[i];
    return C;
  };
  REP(n, N) {
    T d = S[n];
    REP(i, L) d += C_rev[i + 1] * S[n - 1 - i];
    if(d == 0) m++;
    else if (2 * L <= n) {
      poly Q = C_rev;
      C_rev = adjust(C_rev, B, d, b, m);
      L = n + 1 - L, B = Q, b = d, m = 1;
    } else C_rev = adjust(C_rev, B, d, b, m++);
  }
  return {L, C_rev};
}

// Calculate $x^N \bmod f(x)$
// Complexity: $O(K^2 \log N)$ ($K$: deg. of $f$)
// (4, [1, -1, -1]) -> [2, 3]
// ( x^4 = (x^2 + x + 2)(x^2 - x - 1) + 3x + 2 )
template<class T>
vector<T> monomial_mod_polynomial(long long N, const vector<T> &f_rev) {
  assert(!f_rev.empty() && f_rev[0] == 1);
  int K = SZ(f_rev) - 1;
  if(!K) return {};
  int D = 64 - __builtin_clzll(N);
  vector<T> ret(K, 0);
  ret[0] = 1;
  auto self_conv = [](vector<T> x) -> vector<T> {
    int d = SZ(x);
    vector<T> ret(d * 2 - 1);
    REP(i, d) {
      ret[i * 2] += x[i] * x[i];
      REP(j, i) ret[i + j] += x[i] * x[j] * 2;
    }
    return ret;
  };
  for(int d = D; d--;) {
    ret = self_conv(ret);
    for(int i = 2 * K - 2; i >= K; i--) {
      REP(j, k) ret[i - j - 1] -= ret[i] * f_rev[j + 1];
    }
    ret.resize(K);
    if (N >> d & 1) {
      vector<T> c(K);
      c[0] = -ret[K - 1] * f_rev[K];
      for(int i = 1; i < K; i++) c[i] = ret[i - 1] - ret[K - 1] * f_rev[K - i];
      ret = c;
    }
  }
  return ret;
}

// Guess k-th element of the sequence, assuming linear recurrence
template<class T>
T guess_kth_term(const vector<T>& a, long long k) {
  assert(k >= 0);
  if(k < 1LL * SZ(a)) return a[k];
  auto f = BM<T>(a).second;
  auto g = monomial_mod_polynomial<T>(k, f);
  T ret = 0;
  REP(i, SZ(g)) ret += g[i] * a[i];
  return ret;
}
```

6.3 Chinese-Remainder

```
// (rem, mod) {0, 0} for no solution
pair<ll, ll> crt(ll r0, ll m0, ll r1, ll m1) {
  r0 = (r0 % m0 + m0) % m0;
  r1 = (r1 % m1 + m1) % m1;
  if(m0 < m1) swap(r0, r1), swap(m0, m1);
  if(m0 % m1 == 0) {
    if(r0 % m1 != r1) return {0, 0};
  }
  ll g, im, qq;
  g = ext_gcd(m0, m1, im, qq);
  ll u1 = (m1 / g);
  if((r1 - r0) % g) return {0, 0};
  ll x = (r1 - r0) / g % u1 * im % u1;
  r0 += x * m0;
  m0 *= u1;
  if(r0 < 0) r0 += m0;
  return {r0, m0};
}
```

6.4 Combination

```
mint binom(int n, int k) {
  if(k < 0 || k > n) return 0;
  return fact[n] * inv_fact[k] * inv_fact[n - k];
}
// a_1 + a_2 + ... + a_n = k, a_i >= 0
mint stars_and_bars(int n, int k) { return binom(k + n - 1, n - 1); }
// number of ways from (0, 0) to (n, m)
mint paths(int n, int m) { return binom(n + m, n); }
mint catalan(int n) { return binom(2 * n, n) - binom(2 * n, n + 1); }
```

6.5 Determinant

```
T det(vector<vector<T>> a) {
  int n = SZ(a);
  T ret = 1;
  REP(i, n) {
    int idx = -1;
    for(int j = i; j < n; j++) {
      if(a[j][i] != 0) {
        idx = j;
        break;
      }
    }
    if(idx == -1) return 0;
    if(i != idx) {
      ret *= T(-1);
      swap(a[i], a[idx]);
    }
    ret *= a[i][i];
    T inv = T(1) / a[i][i];
    REP(j, n) a[i][j] *= inv;
    for(int j = i + 1; j < n; j++) {
      T x = a[j][i];
      if(x == 0) continue;
      for(int k = i; k < n; k++) {
        a[j][k] -= a[i][k] * x;
      }
    }
  }
  return ret;
}
```

6.6 Discrete-Log

```
int discrete_log(int a, int b, int m) {
  if(b == 1 || m == 1) return 0;
  int n = sqrt(m) + 2, e = 1, f = 1, j = 1;
  unordered_map<int, int> A; // becareful
  while(j <= n && (e = f = 1LL * e * a % m) != b) A[1LL * e * b % m] = j++;
  if(e == b) return j;
  if(__gcd(m, e) == __gcd(m, b)) {
    for(int i = 2; i < n + 2; ++i) {
      e = 1LL * e * f % m;
      if(A.find(e) != A.end()) return n * i - A[e];
    }
  }
  return -1;
}
```

6.7 extgcd

```
// ax + by = gcd(a, b)
ll ext_gcd(ll a, ll b, ll& x, ll& y) {
  if(b == 0) {
    x = 1, y = 0;
    return a;
  }
  ll x1, y1;
  ll g = ext_gcd(b, a % b, x1, y1);
  x = y1, y = x1 - (a / b) * y1;
  return g;
}
```

6.8 Floor-Sum

```
// sum_{i = 0}^{n - 1} floor((ai + b) / c) in O(a + b + c + n)
ll floor_sum(ll n, ll a, ll b, ll c) {
  assert(0 <= n && n < (1LL << 32));
  assert(1 <= c && c < (1LL << 32));
  ull ans = 0;
  if(a < 0) {
    ull a2 = (a % c + c) % c;
    ans -= 1ULL * n * (n - 1) / 2 * ((a2 - a) / c);
    a = a2;
  }
  if(b < 0) {
    ull b2 = (b % c + c) % c;
    ans -= 1ULL * n * ((b2 - b) / c);
    b = b2;
  }
  ull N = n, C = c, A = a, B = b;
  while(true) {
    if(A >= C) {
      ans += N * (N - 1) / 2 * (A / C);
      A %= C;
    }
    if(B >= C) {
      ans += N * (B / C);
      B %= C;
    }
    ull y_max = A * N + B;
    if(y_max < C) break;
    N = y_max / C, B = y_max % C;
    swap(C, A);
  }
  return ans;
}
```

6.9 FWHT

```
#define ppc __builtin_popcount
template<class T, class F>
void fwht(vector<T>& a, F f) {
  int n = SZ(a);
  assert(ppc(n) == 1);
  for(int i = 1; i < n; i <<= 1) {
    for(int j = 0; j < n; j += i << 1) {
      REP(k, i) f(a[j + k], a[i + j + k]);
    }
  }
}
template<class T>
void or_transform(vector<T>& a, bool inv) { fwht(a, [&](T& x, T& y) { y += x * (inv ? -1 : +1); }) }
template<class T>
void and_transform(vector<T>& a, bool inv) { fwht(a, [&](T& x, T& y) { x += y * (inv ? -1 : +1); }); }
template<class T>
void xor_transform(vector<T>& a, bool inv) {
  fwht(a, [](T& x, T& y) {
    T z = x + y;
    y = x - y;
    x = z;
  });
  if(inv) {
    T z = T(1) / T(SZ(a));
    for(auto& x : a) x *= z;
  }
}
template<class T>
vector<T> convolution(vector<T> a, vector<T> b) {
  assert(SZ(a) == SZ(b));
  transform(a, false), transform(b, false);
  REP(i, SZ(a)) a[i] *= b[i];
  transform(a, true);
  return a;
}
template<class T>
vector<T> subset_convolution(const vector<T>& f, const vector<T>& g) {
  assert(SZ(f) == SZ(g));
  int n = SZ(f);
  assert(ppc(n) == 1);
  const int lg = __lg(n);
  vector<vector<T>> fhat(lg + 1, vector<T>(n)), ghat(fhat);
  REP(i, n) fhat[ppc(i)][i] = f[i], ghat[ppc(i)][i] = g[i];
  REP(i, lg + 1) or_transform(fhat[i], false), or_transform(ghat[i], false);
  vector<vector<T>> h(lg + 1, vector<T>(n));
  REP(m, n) REP(i, lg + 1) REP(j, i + 1) h[i][m] += fhat[j][m] * ghat[i - j][m];
  REP(i, lg + 1) or_transform(h[i], true);
  vector<T> res(n);
  REP(i, n) res[i] = h[ppc(i)][i];
  return res;
}
```

6.10 Gauss-Jordan

```
int GaussJordan(vector<vector<ld>>& a) {
  // -1 no sol, 0 inf sol
  int n = SZ(a);
  REP(i, n) assert(SZ(a[i]) == n + 1);
  REP(i, n) {
    int p = i;
    REP(j, n) {
      if(j < i && abs(a[j][j]) > EPS)
        continue;
      if(abs(a[j][i]) > abs(a[p][i])) p = j;
    }
    REP(j, n + 1) swap(a[i][j], a[p][j]);
    if(abs(a[i][i]) <= EPS) continue;
    REP(j, n) {
      if(i == j) continue;
      ld delta = a[j][i] / a[i][i];
      FOR(k, i, n + 1) a[j][k] -= delta * a[i][k];
    }
  }
  bool ok = true;
  REP(i, n) {
    if(abs(a[i][i]) <= EPS) {
      if(abs(a[i][n]) > EPS) return -1;
      ok = false;
    }
  }
  return ok;
}
```

6.11 GCD-Convolution

```
// 2, 3, 5, 7, ...
vector<int> prime_enumerate(int N) {
  vector<bool> sieve(N / 3 + 1, 1);
  for(int p = 5, d = 4, i = 1, sqn = sqrt(N); p <= sqn; p += d = 6 - d, i++) {
    if(!sieve[i]) continue;
    for(int q = p * p / 3, r = d * p / 3 + (d * p % 3 == 2), s = 2 * p; q < SZ(sieve); q += r = s - r) sieve[q] = 0;
  }
  vector<int> ret{2, 3};
  for(int p = 5, d = 4, i = 1; p <= N; p += d = 6 - d, i++) {
    if(sieve[i]) {
      ret.pb(p);
    }
  }
  while(SZ(ret) && ret.back() > N) ret.pop_back();
  return ret;
}
struct divisor_transform {
  template<class T>
  static void zeta_transform(vector<T>& a) {
    int n = a.size() - 1;
    for(auto p : prime_enumerate(n)) {
      for(int i = 1; i * p <= n; i++) {
        a[i * p] += a[i];
      }
    }
  }
  template<class T>
  static void mobius_transform(vector<T>& a) {
    int n = a.size() - 1;
    for(auto p : prime_enumerate(n)) {
      for(int i = n / p; i > 0; i--) {
        a[i * p] -= a[i];
      }
    }
  }
};
struct multiple_transform {
  template<class T>
  static void zeta_transform(vector<T>& a) {
    int n = a.size() - 1;
    for(auto p : prime_enumerate(n)) {
      for(int i = n / p; i > 0; i--) {
        a[i] += a[i * p];
      }
    }
  }
  template<class T>
  static void mobius_transform(vector<T>& a) {
    int n = a.size() - 1;
    for(auto p : prime_enumerate(n)) {
      for(int i = 1; i * p <= n; i++) {
        a[i] -= a[i * p];
      }
    }
  }
};
// lcm: multiple -> divisor
template<class T>
vector<T> gcd_convolution(const vector<T>& a, const vector<T>& b) {
  assert(a.size() == b.size());
  auto f = a, g = b;
  multiple_transform::zeta_transform(f);
  multiple_transform::zeta_transform(g);
  REP(i, SZ(f)) f[i] *= g[i];
  multiple_transform::mobius_transform(f);
  return f;
}
```

6.12 Int-Div

```
ll floor_div(ll a, ll b) {
  return a/b - ((a^b) < 0 && a%b != 0);
}
ll ceil_div(ll a, ll b) {
  return a/b + ((a^b) > 0 && a%b != 0);
}
```

6.13 Linear-Sieve

```
vi primes, least = {0, 1}, phi, mobius;
void LinearSieve(int n) {
  least = phi = mobius = vi(n + 1);
  mobius[1] = 1;
  for(int i = 2; i <= n; i++) {
    if(!least[i]) {
      least[i] = i;
      primes.pb(i);
```

```
      phi[i] = i - 1;
      mobius[i] = -1;
    }
    for(auto j : primes) {
      if(i * j > n) break;
      least[i * j] = j;
      if(i % j == 0) {
        mobius[i * j] = 0;
        phi[i * j] = phi[i] * j;
        break;
      } else {
        mobius[i * j] = -mobius[i];
        phi[i * j] = phi[i] * phi[j];
      }
    }
  }
}
```

6.14 Miller-Rabin

```
bool is_prime(ll n, vector<ll> x) {
  ll d = n - 1;
  d >>= __builtin_ctzll(d);
  for(auto a : x) {
    if(n <= a) break;
    ll t = d, y = 1, b = t;
    while(b) {
      if(b & 1) y = i128(y) * a % n;
      a = i128(a) * a % n;
      b >>= 1;
    }
    while(t != n - 1 && y != 1 && y != n - 1) {
      y = i128(y) * y % n;
      t <<= 1;
    }
    if(y != n - 1 && t % 2 == 0) return false;
  }
  return true;
}
bool is_prime(ll n) {
  if(n <= 1) return false;
  if(n % 2 == 0) return n == 2;
  if(n < (1LL << 30)) return is_prime(n, {2, 7, 61});
  return is_prime(n, {2, 325, 9375, 28178, 450775, 9780504, 1795265022});
}
```

6.15 Min-of-Mod-of-Linear

```
// \min{Ax + B (mod M) | 0 <= x < N}
int min_of_mod_of_linear(int n, int m, int a, int b) {
  ll v = floor_sum(n, m, a, b);
  int l = -1, r = m - 1;
  while(r - l > 1) {
    int k = (l + r) / 2;
    if(floor_sum(n, m, a, b + (m - 1 - k)) < v + n) r = k;
    else l = k;
  }
  return r;
}
```

6.16 Mod-Inv

```
int inv(int a) {
  if(a < N) return inv[a];
  if(a == 1) 1;
  return (MOD - 1LL * (MOD / a) * inv(MOD % a) % MOD) % MOD;
}
vi mod_inverse(int m, int n = -1) {
  assert(n < m);
  if(n == -1) n = m - 1;
  vi inv(n + 1);
  inv[0] = inv[1] = 1;
  for(int i = 2; i <= n; i++) inv[i] = m - 1LL * (m / i) * inv[m % i] % m;
  return inv;
}
```

6.17 Pollard-Rho

```
void PollardRho(map<ll, int>& mp, ll n) {
  if(n == 1) return;
  if(is_prime(n)) return mp[n]++, void();
  if(n % 2 == 0) {
    mp[2] += 1;
    PollardRho(mp, n / 2);
    return;
  }
  ll x = 2, y = 2, d = 1, p = 1;
  #define f(x, n, p) ((i128(x) * x % n + p) % n)
  while(true) {
    if(d != 1 && d != n) {
      PollardRho(mp, d);
      PollardRho(mp, n / d);
      return;
    }
    p += (d == n);
    x = f(x, n, p), y = f(f(y, n, p), n, p);
    d = __gcd(abs(x - y), n);
  }
  #undef f
}

vector<ll> get_divisors(ll n) {
  if(n == 0) return {};
  map<ll, int> mp;
  PollardRho(mp, n);
  vector<pair<ll, int>> v(all(mp));
  vector<ll> res;
  auto f = [&](auto f, int i, ll x) -> void {
    if(i == sz(v)) {
      res.pb(x);
      return;
    }
    for(int j = v[i].second; ; j--) {
      f(f, i + 1, x);
      if(j == 0) break;
      x *= v[i].first;
    }
  };
  f(f, 0, 1);
  sort(all(res));
  return res;
}
```

6.18 Primes

```
/* 12721 13331 14341 75577 123457 222557 556679 999983 1097774749 1076767633 100102021 999997771 1001010013 1000512343 987654361 999991231 999888733 98789101 987777733 999991921 1010101333 1010102101 1000000000039 1000000000000037 2305843009213693951 4611686018427387847 9223372036854775783 18446744073709551557 */
```

6.19 Triangle

```
// Counts x, y >= 0 such that Ax + By <= C. Requires A, B > 0. Runs in log time.
// Also representable as sum_{0 <= x <= C / A} floor((C - Ax) / B + 1).
ll count_triangle(ll A, ll B, ll C) {
  if(C < 0) return 0;
  if(A < B) swap(A, B);
  ll m = C / A, k = A / B;
  ll h = (C - m * A) / B + 1;
  return m * (m + 1) / 2 * k + (m + 1) * h + count_triangle(B, A - k * B, C - B * (k * m + h));
}

// Counts 0 <= x < RA, 0 <= y < RB such that Ax + By <= C. Requires A, B > 0.
ll count_triangle_rectangle_intersection(ll A, ll B, ll C, ll RA, ll RB) {
  if(C < 0 || RA <= 0 || RB <= 0) return 0;
  if(C >= A * (RA - 1) + B * (RB - 1)) return RA * RB;
  return count_triangle(A, B, C) - count_triangle(A, B, C - A * RA) - count_triangle(A, B, C - B * RB);
}
```

6.20 Xor-Basis

```
template<int B>
struct xor_basis {
  using T = long long;
  bool zero = false, change = false;
  int cnt = 0;
  array<T, B> p = {};
  vector<T> d;
  void insert(T x) {
    IREP(i, B) {
      if(x >> i & 1) {
        if(!p[i]) {
          p[i] = x, cnt++;
          change = true;
          return;
        } else x ^= p[i];
      }
    }
    if(!zero) zero = change = true;
  }
  T get_min() {
    if(zero) return 0;
    REP(i, B) if(p[i]) return p[i];
  }
  T get_max() {
    T ans = 0;
    IREP(i, B) ans = max(ans, ans ^ p[i]);
    return ans;
  }
  T get_kth(long long k) {
    k++;
    if(k == 1 && zero) return 0;
    k -= zero;
    if(k >= (1LL << cnt)) return -1;
    update();
    T ans = 0;
    REP(i, SZ(d)) if(k >> i & 1) ans ^= d[i];
    return ans;
  }
  bool contains(T x) {
    if(x == 0) return zero;
    IREP(i, B) if(x >> i & 1) x ^= p[i];
    return x == 0;
  }
  void merge(const xor_basis& other) { REP(i, B) if(other.p[i]) insert(other.p[i]); }
  void update() {
    if(!change) return;
    change = false;
    d.clear();
    REP(j, B) IREP(i, j) if(p[j] >> i & 1) p[j] ^= p[i];
    REP(i, B) if(p[i]) d.pb(p[i]);
  }
};
```

6.21 估計值

- Estimation
  - The number of divisors of n is at most around 100 for $n < 5e4$, 500 for $n < 1e7$, 2000 for $n < 1e10$, 200000 for $n < 1e19$.

- The number of ways of writing n as a sum of positive integers, disregarding the order of the summands. $1, 1, 2, 3, 5, 7, 11, 15, 22, 30$ for $n = 0 \sim 9$, 627 for $n = 20$, $\sim 2e5$ for $n = 50$, $\sim 2e8$ for $n = 100$.
- Total number of partitions of n distinct elements: $B(n) = 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, 190899322, \ldots$.

6.22 定理

- Cramer's rule

$$\begin{matrix} ax + by = e \\ cx + dy = f \end{matrix} \Rightarrow \begin{matrix} x = \dfrac{ed - bf}{ad - bc} \\ y = \dfrac{af - ec}{ad - bc} \end{matrix}$$

- Vandermonde's Identity

$$C(n + m, k) = \sum_{i=0}^{k} C(n, i)C(m, k - i)$$

- Burnside's Lemma

  Let us calculate the number of necklaces of n pearls, where each pearl has m possible colors. Two necklaces are symmetric if they are similar after rotating them. There are n ways to change the position of a necklace, because we can rotate it $0, 1, \ldots, n_1$ steps clockwise. If the number of steps is 0, all m^n necklaces remain the same, and if the number of steps is 1, only the m necklaces where each pearl has the same color remain the same. More generally, when the number of steps is k, a total of $m^{\gcd(k,n)}$ necklaces remain the same. The reason for this is that blocks of pearls of size $\gcd(k, n)$ will replace each other. Thus, according to Burnside′s lemma, the number of necklaces is $\sum_{i=0}^{n-1} \frac{m^{\gcd(i,n)}}{n}$. For example, the number of necklaces of length 4 with 3 colors is $\frac{3^4+3+3^2+3}{4} = 24$

- Lindstr□m–Gessel–Viennot Lemma

  定義

  $\omega(P)$ 表示 P 這條路徑上所有邊的邊權之積。(路徑計數時，可以將邊權都設為 1) (事實上，邊權可以為生成函數) $e(u, v)$ 表示 u 到 v 的 ** 每一條 ** 路徑 P 的 $\omega(P)$ 之和，即 $e(u, v) = \sum_{P:u\to v} \omega(P)$。起點集合 A，是有向無環圖點集的一個子集，大小為 n。終點集合 B，也是有向無環圖點集的一個子集，大小也為 n。一組 $A \to B$ 的不相交路徑 S：S_i 是一條從 A_i 到 $B_{\sigma(S)_i}$ 的路徑 ($\sigma(S)$ 是一個排列)，對於任何 $i \neq j$，S_i 和 S_j 沒有公共頂點。$t(\sigma)$ 表示排列 σ 的逆序對個數。

$$M = \begin{bmatrix} e(A_1, B_1) & e(A_1, B_2) & \cdots & e(A_1, B_n) \\ e(A_2, B_1) & e(A_2, B_2) & \cdots & e(A_2, B_n) \\ \vdots & \vdots & \ddots & \vdots \\ e(A_n, B_1) & e(A_n, B_2) & \cdots & e(A_n, B_n) \end{bmatrix}$$

$$\det(M) = \sum_{S:A\to B} (-1)^{t(\sigma(S))} \prod_{i=1}^{n} \omega(S_i)$$

其中 $\sum_{S:A\to B}$ 表示滿足上文要求的 $A \to B$ 的每一組不相交路徑 S。

- Kirchhoff's Theorem

  Denote L be a $n \times n$ matrix as the Laplacian matrix of graph G, where $L_{ii} = d(i)$, $L_{ij} = -c$ where c is the number of edge (i, j) in G.

  - The number of undirected spanning in G is $|\det(\tilde{L}_{11})|$.
  - The number of directed spanning tree rooted at r in G is $|\det(\tilde{L}_{rr})|$.

- Tutte's Matrix

  Let D be a $n \times n$ matrix, where $d_{ij} = x_{ij}$ (x_{ij} is chosen uniformly at random) if $i < j$ and $(i, j) \in E$, otherwise $d_{ij} = -d_{ji}$. $\frac{rank(D)}{2}$ is the maximum matching on G.

- Cayley's Formula

  - Given a degree sequence $d_1, d_2, \ldots, d_n$ for each labeled vertices, there are $\frac{(n-2)!}{(d_1-1)!(d_2-1)!\cdots(d_n-1)!}$ spanning trees.
  - Let $T_{n,k}$ be the number of labeled forests on n vertices with k components, such that vertex $1, 2, \ldots, k$ belong to different components. Then $T_{n,k} = kn^{n-k-1}$.

- Erd□s–Gallai theorem

  A sequence of nonnegative integers $d_1 \geq \cdots \geq d_n$ can be represented as the degree sequence of a finite simple graph on n vertices if and only if $d_1 + \cdots + d_n$ is even and $\sum_{i-1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k)$ holds for every $1 \leq k \leq n$.

- Gale–Ryser theorem

  A pair of sequences of nonnegative integers $a_1 \geq \cdots \geq a_n$ and $b_1, \ldots, b_n$ is bigraphic if and only if $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$ and $\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{n} \min(b_i, k)$ holds for every $1 \leq k \leq n$.

- Fulkerson–Chen–Anstee theorem

  A sequence $(a_1, b_1), \ldots, (a_n, b_n)$ of nonnegative integer pairs with $a_1 \geq \cdots \geq a_n$ is digraphic if and only if $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$ and $\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} \min(b_i, k-1) + \sum_{i=k+1}^{n} \min(b_i, k)$ holds for every $1 \leq k \leq n$.

- M□bius inversion formula

  - $f(n) = \sum_{d|n} g(d) \Leftrightarrow g(n) = \sum_{d|n} \mu(d) f(\frac{n}{d})$
  - $f(n) = \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu(\frac{d}{n}) f(d)$

- Spherical cap

  - A portion of a sphere cut off by a plane.
  - r: sphere radius, a: radius of the base of the cap, h: height of the cap, θ: $\arcsin(a/r)$.
  - Volume $= \pi h^2(3r - h)/3 = \pi h(3a^2 + h^2)/6 = \pi r^3(2 + \cos\theta)(1 - \cos\theta)^2/3$.
  - Area $= 2\pi rh = \pi(a^2 + h^2) = 2\pi r^2(1 - \cos\theta)$.

6.23 數字

- Bernoulli numbers

  $B_0 - 1, B_1^{\pm} = \pm\frac{1}{2}, B_2 = \frac{1}{6}, B_3 = 0$

  $\sum_{j=0}^{m} \binom{m+1}{j} B_j = 0$, EGF is $B(x) = \frac{x}{e^x - 1} = \sum_{n=0}^{\infty} B_n \frac{x^n}{n!}$.

  $$S_m(n) = \sum_{k=1}^{n} k^m = \frac{1}{m+1} \sum_{k=0}^{m} \binom{m+1}{k} B_k^{+} n^{m+1-k}$$

- Stirling numbers of the second kind Partitions of n distinct elements into exactly k groups.

  $S(n, k) = S(n-1, k-1) + kS(n-1, k)$, $S(n, 1) = S(n, n) = 1$

  $S(n, k) = \frac{1}{k!} \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} i^n$

  $x^n = \sum_{i=0}^{n} S(n, i)(x)_i$

- Pentagonal number theorem

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

- Catalan numbers

  $C_n^{(k)} = \frac{1}{(k-1)n + 1} \binom{kn}{n}$

  $C^{(k)}(x) = 1 + x[C^{(k)}(x)]^k$

- Eulerian numbers

  Number of permutations $\pi \in S_n$ in which exactly k elements are greater than the previous element. k j:s s.t. $\pi(j) > \pi(j+1)$, $k+1$ j:s s.t. $\pi(j) \geq j$, k j:s s.t. $\pi(j) > j$.

  $E(n, k) = (n-k)E(n-1, k-1) + (k+1)E(n-1, k)$

  $E(n, 0) = E(n, n-1) = 1$

  $E(n, k) = \sum_{j=0}^{k} (-1)^j \binom{n+1}{j} (k+1-j)^n$

- 次方和

  $\sum_{k=1}^{n} k^3 = (\frac{n(n+1)}{2})^2$

  $\sum_{k=1}^{n} k^4 = \frac{1}{30}(6n^5 + 15n^4 + 10n^3 - n)$

  $\sum_{k=1}^{n} k^5 = \frac{1}{12}(2n^6 + 6n^5 + 5n^4 - n^2)$

  $\sum_{k=1}^{n} k^6 = \frac{1}{42}(6n^7 + 21n^6 + 21n^5 - 7n^3 + n)$

  General form:

  $\sum_{k=1}^{n} k^p = \frac{1}{p+1}(n \sum_{i=1}^{p} (n+1)^i - \sum_{i=2}^{p} \binom{i}{p+1} \sum_{k=1}^{n} k^{p+1-i})$

6.24 歐幾里得類算法

- $m = \lfloor \frac{an+b}{c} \rfloor$
- Time complexity: $O(\log n)$

$$f(a, b, c, n) = \sum_{i=0}^{n} \lfloor \frac{ai + b}{c} \rfloor$$

$$= \begin{cases} \lfloor \frac{a}{c} \rfloor \cdot \frac{n(n+1)}{2} + \lfloor \frac{b}{c} \rfloor \cdot (n+1) & \\ +f(a \bmod c, b \bmod c, c, n), & a \geq c \vee b \\ 0, & n < 0 \vee a \\ nm - f(c, c-b-1, a, m-1), & \text{otherwise} \end{cases}$$

$$g(a, b, c, n) = \sum_{i=0}^{n} i \lfloor \frac{ai + b}{c} \rfloor$$

$$= \begin{cases} \lfloor \frac{a}{c} \rfloor \cdot \frac{n(n+1)(2n+1)}{6} + \lfloor \frac{b}{c} \rfloor \cdot \frac{n(n+1)}{2} \\ +g(a \bmod c, b \bmod c, c, n), \\ 0, \\ \frac{1}{2} \cdot (n(n+1)m - f(c, c-b-1, a, m-1) \\ -h(c, c-b-1, a, m-1)), \end{cases}$$

$$h(a, b, c, n) = \sum_{i=0}^{n} \lfloor \frac{ai + b}{c} \rfloor^2$$

$$= \begin{cases} \lfloor \frac{a}{c} \rfloor^2 \cdot \frac{n(n+1)(2n+1)}{6} + \lfloor \frac{b}{c} \rfloor^2 \cdot (n+1) \\ +\lfloor \frac{a}{c} \rfloor \cdot \lfloor \frac{b}{c} \rfloor \cdot n(n+1) \\ +h(a \bmod c, b \bmod c, c, n) \\ +2\lfloor \frac{a}{c} \rfloor \cdot g(a \bmod c, b \bmod c, c, n) \\ +2\lfloor \frac{b}{c} \rfloor \cdot f(a \bmod c, b \bmod c, c, n), \\ 0, \\ nm(m+1) - 2g(c, c-b-1, a, m-1) \\ -2f(c, c-b-1, a, m-1) - f(a, b, c, n), \end{cases}$$

6.25 生成函數

- Ordinary Generating Function $A(x) = \sum_{i \geq 0} a_i x^i$
  - $A(rx) \Rightarrow r^n a_n$
  - $A(x) + B(x) \Rightarrow a_n + b_n$
  - $A(x)B(x) \Rightarrow \sum_{i=0}^{n} a_i b_{n-i}$
  - $A(x)^k \Rightarrow \sum_{i_1+i_2+\cdots+i_k=n} a_{i_1} a_{i_2} \ldots a_{i_k}$
  - $xA(x)' \Rightarrow na_n$
  - $\frac{A(x)}{1-x} \Rightarrow \sum_{i=0}^{n} a_i$
- Exponential Generating Function $A(x) = \sum_{i \geq 0} \frac{a_i}{i!} x_i$
  - $A(x) + B(x) \Rightarrow a_n + b_n$
  - $A^{(k)}(x) \Rightarrow a_{n+k}$
  - $A(x)B(x) \Rightarrow \sum_{i=0}^{n} \binom{n}{i} a_i b_{n-i}$
  - $A(x)^k \Rightarrow \sum_{i_1+i_2+\cdots+i_k=n} \binom{n}{i_1, i_2, \ldots, i_k} a_{i_1} a$
  - $xA(x) \Rightarrow na_n$
- Special Generating Function
  - $(1+x)^n = \sum_{i \geq 0} \binom{n}{i} x^i$
  - $\frac{1}{(1-x)^n} = \sum_{i \geq 0} \binom{i}{n-1} x^i$

7 Misc

7.1 fast

```cpp
#pragma GCC optimize("Ofast,no-stack-protector,unroll-loops,fast-math,inline
inline char gc() {
  static const size_t sz = 65536;
  static char buf[sz];
  static char *p = buf, *end = buf;
  if(p == end) end = buf + fread(buf, 1, sz, stdin), p = buf;
  return *p++;
}
```

7.2 for

```cpp
#define FOR(i, begin, end) for(int i = (begin), i##_end_ = (end); i < i##_end_; i++)
#define IFOR(i, begin, end) for(int i = (end) - 1, i##_begin_ = (begin); i >= i##_begin_; i--)
#define REP(i, n) FOR(i, 0, n)
#define IREP(i, n) IFOR(i, 0, n)
```

7.3 next-combination

```cpp
// Example: 1 -> 2, 4 -> 8, 12(1100) -> 17(10001)
ll next_combination(ll comb) {
  ll x = comb & -comb, y = comb + x;
  return ((comb & ~y) / x >> 1) | y;
}
```

7.4 PBDS

```cpp
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
tree<ll, null_type, less<ll>, rb_tree_tag, tree_order_statistics_node_update> st;
// find_by_order order_of_key
// __float128_t
for(int i = bs._Find_first(); i < bs.size(); i = bs._Find_next(i));
```

7.5 python

```python
from decimal import Decimal, getcontext
getcontext().prec = 1000000000
getcontext().Emax = 9999999999
a = pow(Decimal(2), 82589933) - 1
```

7.6 rng

```cpp
inline ull rng() {
  static ull Q = 48763;
  Q ^= Q << 7;
  Q ^= Q >> 9;
  return Q & 0xFFFFFFFFULL;
}
```

7.7 rotate90

```cpp
vector<vector<T>> rotate90(const vector<vector<T>>& a) {
  int n = sz(a), m = sz(a[0]);
  vector<vector<T>> b(m, vector<T>(n));
  REP(i, n) REP(j, m) b[j][i] = a[i][m - 1 - j];
  return b;
}
```

7.8 timer

```cpp
clock_t T1 = clock();
double getCurrentTime() { return (double) (clock() - T1) / CLOCKS_PER_SEC; }
```

8 String

8.1 AC

```cpp
template<int ALPHABET = 26, char MIN_CHAR = 'a'>
struct ac_automaton {
  struct Node {
    int fail = 0, cnt = 0;
    array<int, ALPHABET> go{};
  };
  vector<Node> node;
  vi que;
  int new_node() { return node.eb(), SZ(node) - 1; }
  Node& operator[](int i) { return node[i]; }
  ac_automaton() { new_node(); }
  int insert(const string& s) {
    int p = 0;
    for(char c : s) {
      int v = c - MIN_CHAR;
      if(node[p].go[v] == 0) node[p].go[v] = new_node();
      p = node[p].go[v];
    }
    node[p].cnt++;
    return p;
  }
  void build() {
    que.reserve(SZ(node)); que.pb(0);
    REP(i, SZ(que)) {
      int u = que[i];
      REP(j, ALPHABET) {
        if(node[u].go[j] == 0) node[u].go[j] = node[node[u].fail].go[j];
        else {
          int v = node[u].go[j];
          node[v].fail = (u == 0 ? u : node[node[u].fail].go[j]);
          que.pb(v);
        }
      }
    }
  }
};
```

8.2 KMP

```cpp
// abacbaba -> [0, 0, 1, 0, 0, 1, 2, 3]
vi KMP(const vi& a) {
  int n = SZ(a);
  vi k(n);
  for(int i = 1; i < n; ++i) {
    int j = k[i - 1];
    while(j > 0 && a[i] != a[j]) j = k[j - 1];
    j += (a[i] == a[j]);
    k[i] = j;
  }
  return k;
}
```

8.3 LCP

```cpp
vi lcp(const vi& s, const vi& sa) {
  int n = SZ(s), h = 0;
  vi rnk(n), lcp(n - 1);
  REP(i, n) rnk[sa[i]] = i;
  REP(i, n) {
    h -= (h > 0);
    if(rnk[i] == 0) continue;
    int j = sa[rnk[i] - 1];
    for(; j + h < n && i + h < n; h++) if(s[j + h] != s[i + h]) break;
    lcp[rnk[i] - 1] = h;
  }
  return lcp;
}
```

8.4 manacher

```cpp
// length: (z[i] - (i & 1)) / 2 * 2 + (i & 1)
vi manacher(string t) {
  string s = "&";
  for(char c : t) s.pb(c), s.pb('%');
  int l = 0, r = 0;
  vi z(sz(s));
  REP(i, sz(s)) {
    z[i] = r > i ? min(z[2 * l - i], r - i) : 1;
    while(s[i + z[i]] == s[i - z[i]]) z[i]++;
    if(z[i] + i > r) r = z[i] + 1, l = i;
  }
  return z;
}
```

8.5 rolling-hash

```cpp
const ll M = 911382323, mod = 972663749;
ll Get(vector<ll>& h, int l, int r) {
  if(!l) return h[r]; // p[i] = M^i % mod
  ll ans = (h[r] - h[l - 1] * p[r - l + 1]) % mod;
  return (ans + mod) % mod;
```

```cpp
}
vector<ll> Hash(string s) {
  vector<ll> ans(sz(s));
  ans[0] = s[0];
  for(int i = 1; i < sz(s); i++) ans[i] = (ans[i - 1] * M + s[i]) % mod;
  return ans;
}
```

8.6 SAIS

```cpp
// mississippi
// 10 7 4 1 0 9 8 6 3 5 2
vi SAIS(string a) {
  #define QQ(i, n) for(int i = (n); i >= 0; i--)
  int n = sz(a), m = *max_element(all(a)) + 1;
  vi pos(m + 1), x(m), sa(n), val(n), lms;
  for(auto c : a) pos[c + 1]++;
  REP(i, m) pos[i + 1] += pos[i];
  vector<bool> s(n);
  QQ(i, n - 2) s[i] = a[i] != a[i + 1] ? a[i] < a[i + 1] : s[i + 1];
  auto ind = [&](const vi& ls){
    fill(all(sa), -1);
    auto L = [&](int i) { if(i >= 0 && !s[i]) sa[x[a[i]]++] = i; };
    auto S = [&](int i) { if(i >= 0 && s[i]) sa[--x[a[i]]] = i; };
    REP(i, m) x[i] = pos[i + 1];
    QQ(i, sz(ls) - 1) S(ls[i]);
    REP(i, m) x[i] = pos[i];
    L(n - 1);
    REP(i, n) L(sa[i] - 1);
    REP(i, m) x[i] = pos[i + 1];
    QQ(i, n - 1) S(sa[i] - 1);
  };
  auto ok = [&](int i) { return i == n || (!s[i - 1] && s[i]); };
  auto same = [&](int i,int j) {
    do {
      if(a[i++] != a[j++]) return false;
    } while(!ok(i) && !ok(j));
    return ok(i) && ok(j);
  };
  for(int i = 1; i < n; i++) if(ok(i)) lms.pb(i);
  ind(lms);
  if(sz(lms)) {
    int p = -1, w = 0;
    for(auto v : sa) if(v && ok(v)) {
      if(p != -1 && same(p, v)) w--;
      val[p = v] = w++;
    }
    auto b = lms;
    for(auto& v : b) v = val[v];
    b = SAIS(b);
    for(auto& v : b) v = lms[v];
    ind(b);
  }
  return sa;
}
```

8.7 SAM

```cpp
// cnt 要先用 bfs 往回推, 第一次出現的位置是 state.first_pos - |S| + 1
struct Node { int go[26], len, link, cnt, first_pos; };
Node SA[N]; int sz;
void sa_init() { SA[0].link = -1, SA[0].len = 0, sz = 1; }
int sa_extend(int p, int c) {
  int u = sz++;
  SA[u].first_pos = SA[u].len = SA[p].len + 1;
  SA[u].cnt = 1;
  while(p != -1 && SA[p].go[c] == 0) {
    SA[p].go[c] = u;
    p = SA[p].link;
  }
  if(p == -1) {
    SA[u].link = 0;
    return u;
  }
  int q = SA[p].go[c];
  if(SA[p].len + 1 == SA[q].len) {
    SA[u].link = q;
    return u;
  }
  int x = sz++;
  SA[x] = SA[q];
  SA[x].cnt = 0;
  SA[x].len = SA[p].len + 1;
  SA[q].link = SA[u].link = x;
  while(p != -1 && SA[p].go[c] == q) {
    SA[p].go[c] = x;
    p = SA[p].link;
  }
  return u;
}
```

8.8 smallest-rotation

```cpp
string small_rot(string s) {
  int n = sz(s), i = 0, j = 1;
  s += s;
  while(i < n && j < n) {
    int k = 0;
    while(k < n && s[i + k] == s[j + k]) k++;
    if(s[i + k] <= s[j + k]) j += k + 1;
    else i += k + 1;
    if(i == j) j++;
  }
  int ans = i < n ? i : j;
  return s.substr(ans, n);
}
```

8.9 Z

```cpp
// abacbaba -> [0, 0, 1, 0, 0, 3, 0, 1]
vi z_algorithm(const vi& a) {
  int n = sz(a);
  vi z(n);
  for(int i = 1, j = 0; i < n; ++i) {
    if(i <= j + z[j]) z[i] = min(z[i - j], j + z[j] - i);
    while(i + z[i] < n && a[i + z[i]] == a[z[i]]) z[i]++;
    if(i + z[i] > j + z[j]) j = i;
  }
  return z;
}
```

ACM ICPC Judge Test - NTHU SplayTreap

C++ Resource Test

```cpp
#include <bits/stdc++.h>
using namespace std;

namespace system_test {

const size_t KB = 1024;
const size_t MB = KB * 1024;
const size_t GB = MB * 1024;

size_t block_size, bound;
void stack_size_dfs(size_t depth = 1) {
  if (depth >= bound)
    return;
  int8_t ptr[block_size]; // 若無法編譯將
      block_size 改成常數
  memset(ptr, 'a', block_size);
  cout << depth << endl;
  stack_size_dfs(depth + 1);
}

void stack_size_and_runtime_error(size_t
    block_size, size_t bound = 1024) {
  system_test::block_size = block_size;
  system_test::bound = bound;
  stack_size_dfs();
}

double speed(int iter_num) {
  const int block_size = 1024;
  volatile int A[block_size];
  auto begin = chrono::high_resolution_clock
      ::now();
  while (iter_num--)
    for (int j = 0; j < block_size; ++j)
      A[j] += j;
  auto end = chrono::high_resolution_clock::
      now();
  chrono::duration<double> diff = end -
      begin;
  return diff.count();
}

void runtime_error_1() {
  // Segmentation fault
  int *ptr = nullptr;
  *(ptr + 7122) = 7122;
}

void runtime_error_2() {
  // Segmentation fault
  int *ptr = (int *)memset;
  *ptr = 7122;
}

void runtime_error_3() {
  // munmap_chunk(): invalid pointer
  int *ptr = (int *)memset;
  delete ptr;
}

void runtime_error_4() {
  // free(): invalid pointer
  int *ptr = new int[7122];
  ptr += 1;
  delete[] ptr;
}

void runtime_error_5() {
  // maybe illegal instruction
  int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_6() {
  // floating point exception
  volatile int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_7() {
  // call to abort.
  assert(false);
}

} // namespace system_test

#include <sys/resource.h>
void print_stack_limit() { // only work in
    Linux
  struct rlimit l;
  getrlimit(RLIMIT_STACK, &l);
  cout << "stack_size = " << l.rlim_cur << "
      byte" << endl;
}
```