ACM ICPC Team Reference - NTHU SplayTreap

Contents

1 Basic

1.1 vimrc

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

2 Data-Structure

2.1 DLX

```
1 struct DLX {
2   int n, m, tot, ans;
3   vi first, siz, L, R, U, D, col, row, stk;
4   DLX(int _n, int _m) : n(_n), m(_m), tot(_m
      ) {
5     int sz = n * m;
6     first = siz = L = R = U = D = col = row
        = stk = vi(sz);
7     REP(i, m + 1) {
8       L[i] = i - 1, R[i] = i + 1;
9       U[i] = D[i] = i;
10    }
11    L[0] = m, R[m] = 0;
12  }
13  void insert(int r, int c) {
14    r++, c++;
15    col[++tot] = c, row[tot] = r, ++siz[c];
16    D[tot] = D[c], U[D[c]] = tot, U[tot] = c
        , D[c] = tot;
17    if(!first[r]) first[r] = L[tot] = R[tot]
        = tot;
18    else {
19      L[R[tot] = R[first[r]]] = tot;
20      R[L[tot] = first[r]] = tot;
21    }
22  }
23  #define TRAV(i, X, j) for(i = X[j]; i != j
      ; i = X[i])
24  void remove(int c) {
25    int i, j;
26    L[R[c]] = L[c], R[L[c]] = R[c];
27    TRAV(i, D, c) TRAV(j, R, i) {
28      D[U[D[j]] = U[j]] = D[j];
29      siz[col[j]]--;
30    }
31  }
32  void recover(int c) {
33    int i, j;
34    TRAV(i, U, c) TRAV(j, L, i) {
35      U[D[j]] = D[U[j]] = j;
36      siz[col[j]]++;
37    }
38    L[R[c]] = R[L[c]] = c;
39  }
40  bool dance(int dep) {
41    if(!R[0]) return ans = dep, true;
42    int i, j, c = R[0];
43    TRAV(i, R, 0) if(siz[i] < siz[c]) c = i;
44    remove(c);
45    TRAV(i, D, c) {
46      stk[dep] = row[i];
47      TRAV(j, R, i) remove(col[j]);
48      if(dance(dep + 1)) return true;
49      TRAV(j, L, i) recover(col[j]);
50    }
51    recover(c);
52    return false;
53  }
54  vi solve() {
55    if(!dance(1)) return {};
56    return vi(stk.begin() + 1, stk.begin() +
        ans);
57  }
58 };
```

2.2 fast-set

```
1 // Can correctly work with numbers in range
     [0; MAXN]
2 // Supports all std::set operations in O(1)
     on random queries / dense arrays, O(
     log_64(N)) in worst case (sparce array).
3 // Count operation works in O(1) always.
4 template<uint MAXN>
5 class fast_set {
6 private:
7   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;
12  static constexpr ull lb(int x) {
13    if(x == 64) return ULLONG_MAX;
14    return (1ULL << x) - 1;
15  };
16  static const uint SZ = PREF + (MAXN + 63)
       / 64 + 1;
17  ull m[SZ] = {0};
18  inline uint left(uint v) const { return (v
       - 62) * 64; }
19  inline uint parent(uint v) const { return
      v / 64 + 62; }
20  inline void setbit(uint v) { m[v >> 6] |=
      1ULL << (v & 63); }
21  inline void resetbit(uint v) { m[v >> 6]
      &= ~(1ULL << (v & 63)); }
22  inline void getbit(uint v) const { return
      m[v >> 6] >> (v & 63) & 1; }
23  inline ull childs_value(uint v) const {
      return m[left(v) >> 6]; }
24  inline int left_go(uint x, const uint c)
      const {
25    const ull rem = x & 63;
26    uint bt = PREF * 64 + x;
27    ull num = m[bt >> 6] & lb(rem + c);
28    if(num) return (x ^ rem) | __lg(num);
29    for(bt = parent(bt); bt > 62; bt =
        parent(bt)) {
30      const ull rem = bt & 63;
31      num = m[bt >> 6] & lb(rem);
32      if(num) {
33        bt = (bt ^ rem) | __lg(num);
34        break;
35      }
36    }
37    if(bt == 62) return -1;
38    while(bt < PREF * 64) bt = left(bt) |
        __lg(m[bt - 62]);
39    return bt - PREF * 64;
40  }
41  inline int right_go(uint x, const uint c)
      const {
42    const ull rem = x & 63;
43    uint bt = PREF * 64 + x;
44    ull num = m[bt >> 6] & ~lb(rem + c);
45    if(num) return (x ^ rem) |
        __builtin_ctzll(num);
46    for(bt = parent(bt); bt > 62; bt =
        parent(bt)) {
47      const ull rem = bt & 63;
48      num = m[bt >> 6] & ~lb(rem + 1);
49      if(num) {
50        bt = (bt ^ rem) | __builtin_ctzll(
          num);
51        break;
52      }
53    }
54    if(bt == 62) return -1;
55    while(bt < PREF * 64) bt = left(bt) |
        __builtin_ctzll(m[bt - 62]);
56    return bt - PREF * 64;
57  }
59 public:
60   fast_set() { assert(PREF != 228); setbit
       (62); }
61   bool empty() const {return getbit(63);}
62   void clear() { fill(m, m + SZ, 0); setbit
       (62); }
63   bool count(uint x) const { return m[PREF +
       (x >> 6)] >> (x & 63) & 1; }
64   void insert(uint x) { for(uint v = PREF *
       64 + x; !getbit(v); v = parent(v))
       setbit(v); }
65   void erase(uint x) {
66     if(!getbit(PREF * 64 + x)) return;
67     resetbit(PREF * 64 + x);
68     for(uint v = parent(PREF * 64 + x); v >
         62 && !childs_value(v); v = parent(v
         )) resetbit(v);
69   }
70   int find_next(uint x) const { return
       right_go(x, 0); } // >=
71   int find_prev(uint x) const { return
       left_go(x, 1); } // <=
72 };
```

2.3 lazysegtree

```
1 template<class S,
2         S (*e)(),
3         S (*op)(S, S),
4         class F,
5         F (*id)(),
6         S (*mapping)(F, S),
7         F (*composition)(F, F)>
8 struct lazy_segtree {
9   int n, size, log;
10  vector<S> d; vector<F> lz;
11  void update(int k) { d[k] = op(d[k << 1],
      d[k << 1 | 1]); }
12  void all_apply(int k, F f) {
13    d[k] = mapping(f, d[k]);
14    if(k < size) lz[k] = composition(f, lz[k
       ]);
15  }
16  void push(int k) {
17    all_apply(k << 1, lz[k]);
18    all_apply(k << 1 | 1, lz[k]);
19    lz[k] = id();
20  }
21  lazy_segtree(int _n) : lazy_segtree(vector
      <S>(_n, e())) {}
22  lazy_segtree(const vector<S>& v) : n(sz(v)
      ) {
23    log = __lg(2 * n - 1), size = 1 << log;
24    d.resize(size * 2, e());
25    lz.resize(size, id());
26    REP(i, n) d[size + i] = v[i];
27    for(int i = size - 1; i; i--) update(i);
28  }
29  void set(int p, S x) {
30    p += size;
31    for(int i = log; i; --i) push(p >> i);
32    d[p] = x;
33    for(int i = 1; i <= log; ++i) update(p
        >> i);
34  }
35  S get(int p) {
36    p += size;
37    for(int i = log; i; i--) push(p >> i);
38    return d[p];
39  }
40  S prod(int l, int r) {
41    if(l == r) return e();
42    l += size; r += size;
43    for(int i = log; i; i--) {
44      if(((l >> i) << i) != l) push(l >> i);
45      if(((r >> i) << i) != r) push(r >> i);
46    }
47    S sml = e(), smr = e();
48    while(l < r) {
49      if(l & 1) sml = op(sml, d[l++]);
50      if(r & 1) smr = op(d[--r], smr);
51      l >>= 1;
52      r >>= 1;
53    }
54    return op(sml, smr);
55  }
56  S all_prod() const { return d[1]; }
57  void apply(int p, F f) {
58    p += size;
```

```
59    for(int i = log; i; i--) push(p >> i);
60    d[p] = mapping(f, d[p]);
61    for(int i = 1; i <= log; i++) update(p
          >> i);
62  }
63  void apply(int l, int r, F f) {
64    if(l == r) return;
65    l += size; r += size;
66    for(int i = log; i; i--) {
67      if(((l >> i) << i) != l) push(l >> i);
68      if(((r >> i) << i) != r) push((r - 1)
            >> i);
69    }
70    {
71      int l2 = l, r2 = r;
72      while(l < r) {
73        if(l & 1) all_apply(l++, f);
74        if(r & 1) all_apply(--r, f);
75        l >>= 1;
76        r >>= 1;
77      }
78      l = l2;
79      r = r2;
80    }
81    for(int i = 1; i <= log; i++) {
82      if(((l >> i) << i) != l) update(l >> i
            );
83      if(((r >> i) << i) != r) update((r -
            1) >> i);
84    }
85  }
86  template<class G> int max_right(int l, G g
        ) {
87    assert(0 <= l && l <= n && g(e()));
88    if(l == n) return n;
89    l += size;
90    for(int i = log; i; i--) push(l >> i);
91    S sm = e();
92    do {
93      while(!(l & 1)) l >>= 1;
94      if(!g(op(sm, d[l]))) {
95        while(l < size) {
96          push(l);
97          l <<= 1;
98          if(g(op(sm, d[l]))) sm = op(sm, d[
              l++]);
99        }
100       return l - size;
101     }
102     sm = op(sm, d[l++]);
103   } while((l & -l) != l);
104   return n;
105 }
106 template<class G> int min_left(int r, G g)
         {
107   assert(0 <= r && r <= n && g(e()));
108   if(r == 0) return 0;
109   r += size;
110   for(int i = log; i >= 1; i--) push((r -
          1) >> i);
111   S sm = e();
112   do {
113     r--;
114     while(r > 1 && (r & 1)) r >>= 1;
115     if(!g(op(d[r], sm))) {
116       while(r < size) {
```

```
117         push(r);
118         r = r << 1 | 1;
119         if(g(op(d[r], sm))) sm = op(d[r
                --], sm);
120       }
121       return r + 1 - size;
122     }
123     sm = op(d[r], sm);
124   } while((r & -r) != r);
125   return 0;
126 }
127 };
```

2.4 segtree

```
1  template<class S, S (*e)(), S (*op)(S, S)>
2  struct segtree {
3    int n, size, log;
4    vector<S> st;
5    void update(int v) { st[v] = op(st[v <<
          1], st[v << 1 | 1]); }
6    segtree(int _n) : segtree(vector<S>(_n, e
          ())) {}
7    segtree(const vector<S>& a): n(sz(a)) {
8      log = __lg(2 * n - 1), size = 1 << log;
9      st.resize(size << 1, e());
10     REP(i, n) st[size + i] = a[i];
11     for(int i = size - 1; i; i--) update(i);
12   }
13   void set(int p, S val) {
14     st[p += size] = val;
15     for(int i = 1; i <= log; ++i) update(p
            >> i);
16   }
17   S get(int p) const {
18     return st[p + size];
19   }
20   S prod(int l, int r) const {
21     assert(0 <= l && l <= r && r <= n);
22     S sml = e(), smr = e();
23     l += size, r += size;
24     while(l < r) {
25       if(l & 1) sml = op(sml, st[l++]);
26       if(r & 1) smr = op(st[--r], smr);
27       l >>= 1;
28       r >>= 1;
29     }
30     return op(sml, smr);
31   }
32   S all_prod() const { return st[1]; }
33   template<class F> int max_right(int l, F f
          ) const {
34     assert(0 <= l && l <= n && f(e()));
35     if(l == n) return n;
36     l += size;
37     S sm = e();
38     do {
39       while(~l & 1) l >>= 1;
40       if(!f(op(sm, st[l]))) {
41         while(l < size) {
42           l <<= 1;
43           if(f(op(sm, st[l]))) sm = op(sm,
                st[l++]);
```

```
44         }
45         return l - size;
46       }
47       sm = op(sm, st[l++]);
48     } while((l & -l) != l);
49     return n;
50   }
51   template<class F> int min_left(int r, F f)
           {
52     const {
53     assert(0 <= r && r <= n && f(e()));
54     if(r == 0) return 0;
55     r += size;
56     S sm = e();
57     do {
58       r--;
59       while(r > 1 && (r & 1)) r >>= 1;
60       if(!f(op(st[r], sm))) {
61         while(r < size) {
62           r = r << 1 | 1;
63           if(f(op(st[r], sm))) sm = op(st[r
                --], sm);
64         }
65         return r + 1 - size;
66       }
67       sm = op(st[r], sm);
68     } while((r & -r) != r);
69     return 0;
70   }
71 };
```

2.5 sparse-table

```
1  template<class T, T (*op)(T, T)>
2  struct sparse_table {
3    int n;
4    vector<vector<T>> b;
5    sparse_table(const vector<T>& a) : n(sz(a
          )) {
6      int lg = __lg(n) + 1;
7      b.resize(lg); b[0] = a;
8      for(int j = 1; j < lg; ++j) {
9        b[j].resize(n - (1 << j) + 1);
10       REP(i, n - (1 << j) + 1) b[j][i] = op(
             b[j - 1][i], b[j - 1][i + (1 << (j
             - 1))]);
11     }
12   }
13   T prod(int from, int to) {
14     int lg = __lg(to - from + 1);
15     return op(b[lg][from], b[lg][to - (1 <<
           lg) + 1]);
16   }
17 };
```

2.6 treap

```
1  struct Node {
2    bool rev = false;
3    int sz = 1, pri = rng();
4    Node *l = NULL, *r = NULL, *p = NULL;
```

```
5    // TODO
6  }
7  void pull(Node*& v) {
8    v->sz = 1 + size(v->l) + size(v->r);
9    // TODO
10 }
11 void push(Node*& v) {
12   if(v->rev) {
13     swap(v->l, v->r);
14     if(v->l) v->l->rev ^= 1;
15     if(v->r) v->r->rev ^= 1;
16     v->rev = false;
17   }
18 }
19 Node* merge(Node* a, Node* b) {
20   if(!a || !b) return (a ? a : b);
21   push(a), push(b);
22   if(a->pri > b->pri) {
23     a->r = merge(a->r, b);
24     pull(a); return a;
25   } else {
26     b->l = merge(a, b->l);
27     pull(b); return b;
28   }
29 }
30 pair<Node*, Node*> split(Node* v, int k) {
31   if(!v) return {NULL, NULL};
32   push(v);
33   if(size(v->l) >= k) {
34     auto p = split(v->l, k);
35     if(p.first) p.first->p = NULL;
36     v->l = p.second;
37     pull(v); return {p.first, v};
38   } else {
39     auto p = split(v->r, k - size(v->l) - 1)
             ;
40     if(p.second) p.second->p = NULL;
41     v->r = p.first;
42     pull(v); return {v, p.second};
43   }
44 }
45 int get_position(Node* v) { // 0-indexed
46   int k = (v->l != NULL ? v->l->sz : 0);
47   while(v->p != NULL) {
48     if(v == v->p->r) {
49       k++;
50       if(v->p->l != NULL) k += v->p->l->sz;
51     }
52     v = v->p;
53   }
54   return k;
55 }
```

2.7 動態凸包

```
1  struct line_t {
2    mutable ll k, m, p;
3    bool operator<(const line_t& o) const {
           return k < o.k; }
4    bool operator<(ll x) const { return p < x;
           }
5  };
6  template<bool MAX>
```

```cpp
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)
        ;
  }
};
```

2.8 回滾 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;
    }
  }
}
```

3 Flow-Matching

3.1 Dinic-LowerBound

```cpp
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);
  }
};
```

3.2 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;
  }
};
```

3.3 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 $c_{xyx'y'}$.

3.4 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;
      }
```

```cpp
      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;
  }
};
```

3.5 MCMF

```cpp
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;
```

```cpp
          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};
  }
};
```

3.6 一般圖最大匹配

```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;
```

```cpp
      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;
  }
};
```

3.7 一般圖最小權完美匹配

```cpp
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;
```

```cpp
32        stk.pop_back();
33        return false;
34      }
35      int solve() {
36        for(int i = 0; i < n; i += 2) match[i] =
             i + 1, match[i+1] = i;
37        while(true) {
38          int found = 0;
39          for(int i=0; i<n; i++) dis[i] = onstk[
               i] = 0;
40          for(int i=0; i<n; i++){
41            stk.clear();
42            if(!onstk[i] && SPFA(i)){
43              found = 1;
44              while(stk.size()>=2){
45                int u = stk.back(); stk.pop_back
                     ();
46                int v = stk.back(); stk.pop_back
                     ();
47                match[u] = v;
48                match[v] = u;
49              }
50            }
51          }
52          if(!found) break;
53        }
54        int ans = 0;
55        for(int i=0; i<n; i++) ans += edge[i][
             match[i]];
56        return ans / 2;
57      }
58    }graph;
```

3.8   二分圖最大匹配

```cpp
1  struct bipartite_matching {
2    int n, m; // 二分圖左右人數 (0 ~ n-1), (0
           ~ m-1)
3    vector<vi> g;
4    vi lhs, rhs, dist; // i 與 lhs[i] 配對 (
         lhs[i] == -1 代表沒有配對)
5    bipartite_matching(int _n, int _m) : n(_n)
         , m(_m), g(_n), lhs(_n, -1), rhs(_m,
         -1), dist(_n) {}
6    void add_edge(int u, int v) { g[u].pb(v);
         }
7    void bfs() {
8      queue<int> q;
9      REP(i, n) {
10       if(lhs[i] == -1) {
11         q.push(i);
12         dist[i] = 0;
13       } else {
14         dist[i] = -1;
15       }
16     }
17     while(!q.empty()) {
18       int u = q.front(); q.pop();
19       for(auto v : g[u]) {
20         if(rhs[v] != -1 && dist[rhs[v]] ==
             -1) {
21           dist[rhs[v]] = dist[u] + 1;
```

```cpp
22           q.push(rhs[v]);
23         }
24       }
25     }
26   }
27   bool dfs(int u) {
28     for(auto v : g[u]) {
29       if(rhs[v] == -1) {
30         rhs[lhs[u] = v] = u;
31         return true;
32       }
33     }
34     for(auto v : g[u]) {
35       if(dist[rhs[v]] == dist[u] + 1 && dfs(
             rhs[v])) {
36         rhs[lhs[u] = v] = u;
37         return true;
38       }
39     }
40     return false;
41   }
42   int solve() {
43     int ans = 0;
44     while(true) {
45       bfs();
46       int aug = 0;
47       REP(i, n) if(lhs[i] == -1) aug += dfs(
             i);
48       if(!aug) break;
49       ans += aug;
50     }
51     return ans;
52   }
53 };
```

4   Geometry

4.1   convex-hull

```cpp
1  void convex_hull(vector<P>& dots) {
2    sort(all(dots));
3    vector<P> ans(1, dots[0]);
4    for(int it = 0; it < 2; it++, reverse(all(
         dots))) {
5      for(int i = 1, t = sz(ans); i < sz(dots)
           ; ans.pb(dots[i++])) {
6        while(sz(ans) > t && ori(ans[sz(ans) -
             2], ans.back(), dots[i]) < 0) {
7          ans.ppb();
8        }
9      }
10   }
11   ans.ppb();
12   swap(ans, dots);
13 }
```

4.2   point-in-convex-hull

```cpp
1  int point_in_convex_hull(const vector<P>& a,
         P p) {
2    // -1 ON, 0 OUT, +1 IN
3    // 要先逆時針排序
4    int n = sz(a);
5    if(btw(a[0], a[1], p) || btw(a[0], a[n -
         1], p)) return -1;
6    int l = 0, r = n - 1;
7    while(l <= r) {
8      int m = (l + r) / 2;
9      auto a1 = cross(a[m] - a[0], p - a[0]);
10     auto a2 = cross(a[(m + 1) % n] - a[0], p
           - a[0]);
11     if(a1 >= 0 && a2 <= 0) {
12       auto res = cross(a[(m + 1) % n] - a[m
             ], p - a[m]);
13       return res > 0 ? 1 : (res >= 0 ? -1 :
             0);
14     }
15     if(a1 < 0) r = m - 1;
16     else l = m + 1;
17   }
18   return 0;
19 }
```

4.3   point

```cpp
1  using P = pair<ll, ll>;
2  P operator+(P a, P b) { return P{a.X + b.X,
       a.Y + b.Y}; }
3  P operator-(P a, P b) { return P{a.X - b.X,
       a.Y - b.Y}; }
4  P operator*(P a, ll b) { return P{a.X * b, a
       .Y * b}; }
5  P operator/(P a, ll b) { return P{a.X / b, a
       .Y / b}; }
6  ll dot(P a, P b) { return a.X * b.X + a.Y *
       b.Y; }
7  ll cross(P a, P b) { return a.X * b.Y - a.Y
       * b.X; }
8  ll abs2(P a) { return dot(a, a); }
9  double abs(P a) { return sqrt(abs2(a)); }
10 int sign(ll x) { return x < 0 ? -1 : (x == 0
       ? 0 : 1); }
11 int ori(P a, P b, P c) { return sign(cross(b
       - a, c - a)); }
12 bool collinear(P a, P b, P c) { return sign(
       cross(a - c, b - c)) == 0; }
13 bool btw(P a, P b, P c) {
14   if(!collinear(a, b, c)) return 0;
15   return sign(dot(a - c, b - c)) <= 0;
16 }
17 bool seg_intersect(P a, P b, P c, P d) {
18   int a123 = ori(a, b, c);
19   int a124 = ori(a, b, d);
20   int a341 = ori(c, d, a);
21   int a342 = ori(c, d, b);
22   if(a123 == 0 && a124 == 0) {
23     return btw(a, b, c) || btw(a, b, d) ||
           btw(c, d, a) || btw(c, d, b);
24   }
25   return a123 * a124 <= 0 && a341 * a342 <=
         0;
```

```cpp
26 }
27
28 P intersect(P a, P b, P c, P d) {
29   int a123 = cross(b - a, c - a);
30   int a124 = cross(b - a, d - a);
31   return (d * a123 - c * a124) / (a123 -
         a124);
32 }
33 struct line { P A, B; };
34 P vec(line L) { return L.B - L.A; }
35 P projection(P p, line L) { return L.A + vec
       (L) / abs(vec(L)) * dot(p - L.A, vec(L))
       / abs(vec(L)); }
```

4.4   polar-angle-sort

```cpp
1  bool cmp(P a, P b) {
2    #define ng(k) (sign(k.Y) < 0 || (sign(k.Y)
         == 0 && sign(k.X) < 0))
3    int A = ng(a), B = ng(b);
4    if(A != B) return A < B;
5    if(sign(cross(a, b)) == 0) return abs2(a)
         < abs2(b);
6    return sign(cross(a, b)) > 0;
7  }
```

4.5   最近點對

```cpp
1  const ll INF = 9e18L + 5;
2  vector<P> a;
3  sort(all(a), [](P a, P b) { return a.x < b.x
       ; });
4  ll SQ(ll x) { return x * x; }
5  ll solve(int l, int r) {
6    if(l + 1 == r) return INF;
7    int m = (l + r) / 2;
8    ll midx = a[m].x;
9    ll d = min(solve(l, m), solve(m, r));
10   inplace_merge(a.begin() + l, a.begin() + m
         , a.begin() + r, [](P a, P b) {
11     return a.y < b.y;
12   });
13   vector<P> p;
14   for(int i = l; i < r; ++i) if(SQ(a[i].x -
         midx) < d) p.pb(a[i]);
15   REP(i, sz(p)) {
16     for(int j = i + 1; j < sz(p); ++j) {
17       d = min(d, SQ(p[i].x - p[j].x) + SQ(
             p[i].y - p[j].y));
18       if(SQ(p[i].y - p[j].y) > d) break;
19     }
20   }
21   return d; // 距離平方
22 }
```

5 Graph

5.1 2-SAT

```cpp
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

```cpp
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);
```

```cpp
        }
        return cd;
    };
    int root = build(build, 0);
    return {root, h};
}
```

5.3 HLD

```cpp
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;
```

```cpp
        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);
        }
    }
};
```

5.4 lowlink

```cpp
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);
```

```cpp
        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);
```

```
75    REP(i, n) comp[tecc_id[i]].pb(i);
76    return comp;
77  }
78  vector<vi> bcc_vertices() { // 點雙
79    vector<vi> comp(tvcc_cnt);
80    REP(i, sz(edges)) {
81      comp[tvcc_id[i]].pb(edges[i].first);
82      comp[tvcc_id[i]].pb(edges[i].second);
83    }
84    for(auto& v : comp) {
85      sort(all(v));
86      v.erase(unique(all(v)), v.end());
87    }
88    REP(i, n) if(g[i].empty()) comp.pb({i});
89    return comp;
90  }
91  vector<vi> bcc_edges() {
92    vector<vi> ret(tvcc_cnt);
93    REP(i, sz(edges)) ret[tvcc_id[i]].pb(i);
94    return ret;
95  }
96 };
```

5.5   SCC

```
1  struct SCC {
2    int n;
3    vector<vi> g, h;
4    SCC(int _n) : n(_n), g(_n), h(_n) {}
5    void add_edge(int u, int v) {
6      g[u].pb(v);
7      h[v].pb(u);
8    }
9    vi solve() { // 回傳縮點的編號
10     vi id(n), top;
11     top.reserve(n);
12     #define GO if(id[v] == 0) dfs1(v);
13     function<void(int)> dfs1 = [&](int u) {
14       id[u] = 1;
15       for(auto v : g[u]) GO;
16       top.pb(u);
17     };
18     REP(v, n) GO;
19     fill(all(id), -1);
20     function<void(int, int)> dfs2 = [&](int
          u, int x) {
21       id[u] = x;
22       for(auto v : h[u]) {
23         if(id[v] == -1) {
24           dfs2(v, x);
25         }
26       }
27     };
28     for(int i = n - 1, cnt = 0; i >= 0; --i)
          {
29       int u = top[i];
30       if(id[u] == -1) {
31         dfs2(u, cnt);
32         cnt += 1;
33       }
34     }
35     return id;
```

```
36   }
37 };
```

6   Math

6.1   Aliens

```
1  template<class Func, bool MAX>
2  ll Aliens(ll l, ll r, int k, Func f) {
3    while(l < r) {
4      ll m = l + (r - l) / 2;
5      auto [score, op] = f(m);
6      if(op == k) return score + m * k * (MAX
          ? +1 : -1);
7      if(op < k) r = m;
8      else l = m + 1;
9    }
10   return f(l).first + l * k * (MAX ? +1 :
        -1);
11 }
```

6.2   Berlekamp-Massey

```
1  // - [1, 2, 4, 8, 16] -> (1, [1, -2])
2  // - [1, 1, 2, 3, 5, 8] -> (2, [1, -1, -1])
3  // - [0, 0, 0, 0, 1] -> (5, [1, 0, 0, 0, 0,
       998244352]) (mod 998244353)
4  // - [] -> (0, [1])
5  // - [0, 0, 0] -> (0, [1])
6  // - [-2] -> (1, [1, 2])
7  template<class T>
8  pair<int, vector<T>> BM(const vector<T>& S)
       {
9    using poly = vector<T>;
10   int N = SZ(S);
11   poly C_rev{1}, B{1};
12   int L = 0, m = 1;
13   T b = 1;
14   auto adjust = [](poly C, const poly &B, T
        d, T b, int m) -> poly {
15     C.resize(max(SZ(C), SZ(B) + m));
16     T a = d / b;
17     REP(i, SZ(B)) C[i + m] -= a * B[i];
18     return C;
19   };
20   REP(n, N) {
21     T d = S[n];
22     REP(i, L) d += C_rev[i + 1] * S[n - 1 -
          i];
23     if(d == 0) m++;
24     else if (2 * L <= n) {
25       poly Q = C_rev;
26       C_rev = adjust(C_rev, B, d, b, m);
27       L = n + 1 - L, B = Q, b = d, m = 1;
28     } else C_rev = adjust(C_rev, B, d, b, m
          ++);
29   }
30   return {L, C_rev};
```

```
31 }
32
33 // Calculate $x^N \bmod f(x)$
34 // Complexity: $O(K^2 \log N)$ ($K$: deg. of
       $f$)
35 // (4, [1, -1, -1]) -> [2, 3]
36 // ( x^4 = (x^2 + x + 2)(x^2 - x - 1) + 3x +
       2 )
37 template<class T>
38 vector<T> monomial_mod_polynomial(long long
       N, const vector<T> &f_rev) {
39   assert(!f_rev.empty() && f_rev[0] == 1);
40   int K = SZ(f_rev) - 1;
41   if(!K) return {};
42   int D = 64 - __builtin_clzll(N);
43   vector<T> ret(K, 0);
44   ret[0] = 1;
45   auto self_conv = [](vector<T> x) -> vector
        <T> {
46     int d = SZ(x);
47     vector<T> ret(d * 2 - 1);
48     REP(i, d) {
49       ret[i * 2] += x[i] * x[i];
50       REP(j, i) ret[i + j] += x[i] * x[j] *
            2;
51     }
52     return ret;
53   };
54   for(int d = D; d--;) {
55     ret = self_conv(ret);
56     for(int i = 2 * K - 2; i >= K; i--) {
57       REP(j, k) ret[i - j - 1] -= ret[i] *
            f_rev[j + 1];
58     }
59     ret.resize(K);
60     if (N >> d & 1) {
61       vector<T> c(K);
62       c[0] = -ret[K - 1] * f_rev[K];
63       for(int i = 1; i < K; i++) c[i] = ret[
            i - 1] - ret[K - 1] * f_rev[K - i
            ];
64       ret = c;
65     }
66   }
67   return ret;
68 }
69
70 // Guess k-th element of the sequence,
       assuming linear recurrence
71 template<class T>
72 T guess_kth_term(const vector<T>& a, long
       long k) {
73   assert(k >= 0);
74   if(k < 1LL * SZ(a)) return a[k];
75   auto f = BM<T>(a).second;
76   auto g = monomial_mod_polynomial<T>(k, f);
77   T ret = 0;
78   REP(i, SZ(g)) ret += g[i] * a[i];
79   return ret;
80 }
```

6.3   Chinese-Remainder

```
1  // (rem, mod) {0, 0} for no solution
2  pair<ll, ll> crt(ll r0, ll m0, ll r1, ll m1)
        {
3    r0 = (r0 % m0 + m0) % m0;
4    r1 = (r1 % m1 + m1) % m1;
5    if(m0 < m1) swap(r0, r1), swap(m0, m1);
6    if(m0 % m1 == 0) {
7      if(r0 % m1 != r1) return {0, 0};
8    }
9    ll g, im, qq;
10   g = ext_gcd(m0, m1, im, qq);
11   ll u1 = (m1 / g);
12   if((r1 - r0) % g) return {0, 0};
13   ll x = (r1 - r0) / g % u1 * im % u1;
14   r0 += x * m0;
15   m0 *= u1;
16   if(r0 < 0) r0 += m0;
17   return {r0, m0};
18 }
```

6.4   Discrete-Log

```
1  int discrete_log(int a, int b, int m) {
2    if(b == 1 || m == 1) return 0;
3    int n = sqrt(m) + 2, e = 1, f = 1, j = 1;
4    unordered_map<int, int> A; // becareful
5    while(j <= n && (e = f = 1LL * e * a % m)
          != b) A[1LL * e * b % m] = j++;
6    if(e == b) return j;
7    if(__gcd(m, e) == __gcd(m, b)) {
8      for(int i = 2; i < n + 2; ++i) {
9        e = 1LL * e * f % m;
10       if(A.find(e) != A.end()) return n * i
            - A[e];
11     }
12   }
13   return -1;
14 }
```

6.5   extgcd

```
1  // ax + by = gcd(a, b)
2  ll ext_gcd(ll a, ll b, ll& x, ll& y) {
3    if(b == 0) {
4      x = 1, y = 0;
5      return a;
6    }
7    ll x1, y1;
8    ll g = ext_gcd(b, a % b, x1, y1);
9    x = y1, y = x1 - (a / b) * y1;
10   return g;
11 }
```

6.6 Floor-Sum

```
// sum_0^{n-1} floor((a * i + b) / m) in log
    (n + m + a + b)
ll floor_sum(ll n, ll m, ll a, ll b) {
  ll ans = 0;
  if(a >= m) ans += (n - 1) * n * (a / m) /
      2, a %= m;
  if(b >= m) ans += n * (b / m), b %= m;
  ll y_max = (a * n + b) / m, x_max = (y_max
      * m - b);
  if(y_max == 0) return ans;
  ans += (n - (x_max + a - 1) / a) * y_max;
  return ans + floor_sum(y_max, a, m, (a -
      x_max % a) % a);
}
```

6.7 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;
}
```

6.8 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;
}
```

6.9 Linear-Sieve

```
vi primes, least = {0, 1}, phi, mobius;
void LinearSieve(int n) {
  least = phi = mobius = vi(n + 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];
      }
    }
  }
}
```

6.10 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});
}
```

6.11 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 - 1
      LL * (m / i) * inv[m % i] % m;
  return inv;
}
```

6.12 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;
}
```

6.13 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 */
```

6.14 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: $B(n) = 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, 190899322, \ldots$.

6.15 定理

- Cramer's rule

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

- Vandermonde's Identity

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

- 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 $T_{n,k} = kn^{n-k-1}$.

- Erdős–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 $1 \leq k \leq n$.

- Möbius 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 $= 2\pi rh = \pi(a^2 + h^2) = 2\pi r^2 (1 - \cos\theta)$.

6.16 整數除法

```
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);
}
```

6.17 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.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:

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

6.18 歐幾里得類算法

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

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

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

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

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

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

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

6.19 生成函數

- Ordinary Generating Function $A(x) = \sum_{i \geq 0} a_i x^i$

  - $A(rx) \Rightarrow r^n a_n$
  - $A(x) + B(x) \Rightarrow a_n + b_n$
  - $A(x)B(x) \Rightarrow \sum_{i=0}^{n} a_i b_{n-i}$
  - $A(x)^k \Rightarrow \sum_{i_1 + i_2 + \cdots + i_k = n} a_{i_1} a_{i_2} \cdots a_{i_k}$
  - $xA(x)' \Rightarrow na_n$
  - $\frac{A(x)}{1-x} \Rightarrow \sum_{i=0}^{n} a_i$

- Exponential Generating Function $A(x) = \sum_{i \geq 0} \frac{a_i}{i!} x_i$

  - $A(x) + B(x) \Rightarrow a_n + b_n$
  - $A^{(k)}(x) \Rightarrow a_{n+k}$
  - $A(x)B(x) \Rightarrow \sum_{i=0}^{n} \binom{n}{i} a_i b_{n-i}$
  - $A(x)^k \Rightarrow \sum_{i_1 + i_2 + \cdots + i_k = n} \binom{n}{i_1, i_2, \ldots, i_k} a_{i_1} a_{i_2} \ldots$
  - $xA(x) \Rightarrow na_n$

- Special Generating Function

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

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 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;
}
```

7.3 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));
```

7.4 python

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

7.5 rng

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

7.6 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;
}
```

7.7 timer

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

7.8 矩形覆蓋面積

```cpp
const int N = 2e6 + 5; // [-1e6, 1e6]
int tag[N * 4], seg[N * 4];
void pull(int v, int l, int r) {
  seg[v] = 0;
  if(tag[v] > 0) seg[v] = r - l + 1;
  else if(l < r) seg[v] = seg[v * 2] + seg[v
       * 2 + 1];
}
void update(int ql, int qr, int x, int v =
    1, int l = 0, int r = N - 1) {
  if(ql > r || qr < l) return;
  if(ql <= l && r <= qr) {
    tag[v] += x;
  } else {
    int m = (l + r) / 2;
    update(ql, qr, x, v * 2, l, m);
    update(ql, qr, x, v * 2 + 1, m + 1, r);
  }
  pull(v, l, r);
}
int main() {
  int n; cin >> n;
  vector<array<int, 4>> ev(2 * n);
  REP(i, n) {
    int x, y, x2, y2;
    cin >> x >> y >> x2 >> y2;
    x += N / 2; y += N / 2;
    x2 += N / 2; y2 += N / 2;
    ev[2 * i] = {x, y, y2, +1};
    ev[2 * i + 1] = {x2, y, y2, -1};
  }
  sort(all(ev));
  ll ans = 0, prev = 0;
  REP(i, 2 * n) {
    ans += (ev[i][0] - prev) * seg[1];
    int j = i;
    while(j < 2 * n && ev[i][0] == ev[j][0])
         {
      update(ev[j][1], ev[j][2] - 1, ev[j
          ][3]);
      j++;
    }
    prev = ev[i][0], i = j - 1;
  }
  cout << ans << "\n";
}
```

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) {
```

```cpp
  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] + i, l = i;
  }
  return z;
}
```

8.5 rolling-hash

```
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;
}
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

```
// 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

```
// 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

```
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

```
// 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
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  namespace system_test {
5
6  const size_t KB = 1024;
7  const size_t MB = KB * 1024;
8  const size_t GB = MB * 1024;
9  size_t block_size, bound;
10 void stack_size_dfs(size_t depth = 1) {
11   if (depth >= bound)
12     return;
13   int8_t ptr[block_size]; // 若無法編譯將
                             block_size 改成常數
14   memset(ptr, 'a', block_size);
15   cout << depth << endl;
16   stack_size_dfs(depth + 1);
17 }
18
19 void stack_size_and_runtime_error(size_t
         block_size, size_t bound = 1024) {
20   system_test::block_size = block_size;
21   system_test::bound = bound;
22   stack_size_dfs();
23 }
24
25 double speed(int iter_num) {
26   const int block_size = 1024;
27   volatile int A[block_size];
28   auto begin = chrono::high_resolution_clock
         ::now();
29   while (iter_num--)
30     for (int j = 0; j < block_size; ++j)
31       A[j] += j;
32   auto end = chrono::high_resolution_clock::
         now();
34   chrono::duration<double> diff = end -
         begin;
35   return diff.count();
36 }
37
38 void runtime_error_1() {
39   // Segmentation fault
40   int *ptr = nullptr;
41   *(ptr + 7122) = 7122;
42 }
43
44 void runtime_error_2() {
45   // Segmentation fault
46   int *ptr = (int *)memset;
47   *ptr = 7122;
48 }
49
50 void runtime_error_3() {
51   // munmap_chunk(): invalid pointer
52   int *ptr = (int *)memset;
53   delete ptr;
54 }
55
56 void runtime_error_4() {
57   // free(): invalid pointer
58   int *ptr = new int[7122];
59   ptr += 1;
60   delete[] ptr;
61 }
62 void runtime_error_5() {
63   // maybe illegal instruction
64   int a = 7122, b = 0;
65   cout << (a / b) << endl;
66 }
67
68 void runtime_error_6() {
69   // floating point exception
70   volatile int a = 7122, b = 0;
71   cout << (a / b) << endl;
72 }
73
74 void runtime_error_7() {
75   // call to abort.
76   assert(false);
77 }
78
79 } // namespace system_test
80
81 #include <sys/resource.h>
82 void print_stack_limit() { // only work in
         Linux
83   struct rlimit l;
84   getrlimit(RLIMIT_STACK, &l);
85   cout << "stack_size = " << l.rlim_cur << "
         byte" << endl;
86 }
```