

1 Data-Structure

1.1 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(
  // log64(N)) in worst case (sparse 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 :
8         MAXN <= 4096 ? 1 :
9         MAXN <= 262144 ? 1 + 64 :
10        MAXN <= 16777216 ? 1 + 64 +
11        4096 :
12        MAXN <= 1073741824 ? 1 + 64
13        + 4096 + 262144 : 227) +
14        1;
15     static constexpr ull lb(int x) {
16         if(x == 64) return ULLONG_MAX;
17         return (1ULL << x) - 1;
18     };
19     static const uint SZ = PREF + (MAXN + 63)
20         / 64 + 1;
21     ull m[SZ] = {0};
22     inline uint left(uint v) const { return (v
23         - 62) * 64; }
24     inline uint parent(uint v) const { return
25         v / 64 + 62; }
26     inline void setbit(uint v) { m[v >> 6] |=
27         1ULL << (v & 63); }
28     inline void resetbit(uint v) { m[v >> 6]
29         &= ~(1ULL << (v & 63)); }
30     inline uint getbit(uint v) const { return
31         m[v >> 6] >> (v & 63) & 1; }
32     inline ull childs_value(uint v) const {
33         return m[left(v) >> 6]; }
34     inline int left_go(uint x, const uint c)
35         const {
36         const ull rem = x & 63;
37         uint bt = PREF * 64 + x;
38         ull num = m[bt >> 6] & lb(rem + c);
39         if(num) return (x ^ rem) | __lg(num);
40         for(bt = parent(bt); bt > 62; bt =
41             parent(bt)) {
42             const ull rem = bt & 63;
43             num = m[bt >> 6] & lb(rem + 1);
44             if(num) {
45                 bt = (bt ^ rem) | __builtin_ctzll(
46                     num);
47                 break;
48             }
49             if(bt == 62) return -1;
50             while(bt < PREF * 64) bt = left(bt) |
51                 __builtin_ctzll(m[bt - 62]);
52             return bt - PREF * 64;
53         }
54     public:
55     fast_set() { assert(PREF != 228); setbit
56         (62); }
57     bool empty() const {return getbit(63);}
58     void clear() { fill(m, m + SZ, 0); setbit
59         (62); }
60     bool count(uint x) const { return m[PREF +
61         (x >> 6)] >> (x & 63) & 1; }
62     void insert(uint x) { for(uint v = PREF *
63         64 + x; !getbit(v); v = parent(v))
64         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 >
69             62 && !childs_value(v); v = parent(v)
70             ) resetbit(v);
71     }
72     int find_next(uint x) const { return
73         right_go(x, 0); } // >=
74     int find_prev(uint x) const { return
75         left_go(x, 1); } // <=
76 };

```

1.2 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],
12        d[k << 1 | 1]); }
13    void all_apply(int k, F f) {
14        d[k] = mapping(f, d[k]);
15        if(k < size) lz[k] = composition(f, lz[k]
16        ); }
17 };

```

```

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
22     <S>(_n, e())) {}
23 lazy_segtree(const vector<S>& v) : n(sz(v)
24     ) {
25     log = __lg(2 * n - 1), size = 1 << log;
26     d.resize(size * 2, e());
27     lz.resize(size, id());
28     REP(i, n) d[size + i] = v[i];
29     for(int i = size - 1; i; i--) update(i);
30 }
31 void set(int p, S x) {
32     p += size;
33     for(int i = log; i; --i) push(p >> i);
34     d[p] = x;
35     for(int i = 1; i <= log; ++i) update(p
36         >> i);
37 }
38 S get(int p) {
39     p += size;
40     for(int i = log; i; i--) push(p >> i);
41     return d[p];
42 }
43 S prod(int l, int r) {
44     if(l == r) return e();
45     l += size; r += size;
46     for(int i = log; i; i--) {
47         if(((l >> i) << i) != 1) push(l >> i);
48         if(((r >> i) << i) != r) push(r >> i);
49     }
50     S sm1 = e(), smr = e();
51     while(l < r) {
52         if(l & 1) sm1 = op(sm1, d[l++]);
53         if(r & 1) smr = op(d[--r], smr);
54         l >>= 1;
55         r >>= 1;
56     }
57     return op(sm1, smr);
58 }
59 S all_prod() const { return d[1]; }
60 void apply(int p, F f) {
61     p += size;
62     for(int i = log; i; i--) push(p >> i);
63     d[p] = mapping(f, d[p]);
64     for(int i = 1; i <= log; ++i) update(p
65         >> i);
66 }
67 void apply(int l, int r, F f) {
68     if(l == r) return;
69     l += size; r += size;
70     for(int i = log; i; i--) {
71         if(((l >> i) << i) != 1) push(l >> i);
72         if(((r >> i) << i) != r) push((r - 1)
73             >> i);
74     }
75     {
76         int l2 = l, r2 = r;
77         while(l < r) {
78             if(l & 1) all_apply(l++, f);
79             if(r & 1) all_apply(--r, f);
80             l >>= 1;
81             r >>= 1;
82         }
83     }
84 }

```

```

77     }
78     l = l2;
79     r = r2;
80 }
81 for(int i = 1; i <= log; ++i) {
82     if(((l >> i) << i) != 1) update(l >> i
83         );
84     if(((r >> i) << i) != r) update((r -
85         1) >> i);
86 }
87 }
88 template<class G> int max_right(int l, G g
89     ) {
90     assert(0 <= l && l <= n && g(e()));
91     if(l == n) return n;
92     l += size;
93     for(int i = log; i; i--) push(l >> i);
94     S sm = e();
95     do {
96         while(!(l & 1)) l >>= 1;
97         if(!g(op(sm, d[l]))) {
98             while(l < size) {
99                 push(l);
100                 l <<= 1;
101                 if(g(op(sm, d[l]))) sm = op(sm, d[
102                     l++]);
103             }
104             return l - size;
105         }
106         sm = op(sm, d[l++]);
107         while((l & -l) != 1);
108         return n;
109     }
110     template<class G> int min_left(int r, G g)
111         {
112         assert(0 <= r && r <= n && g(e()));
113         if(r == 0) return 0;
114         r += size;
115         for(int i = log; i >= 1; i--) push((r -
116             1) >> i);
117         S sm = e();
118         do {
119             r--;
120             while(r > 1 && (r & 1)) r >>= 1;
121             if(!g(op(d[r], sm))) {
122                 while(r < size) {
123                     push(r);
124                     r = r << 1 | 1;
125                     if(g(op(d[r], sm))) sm = op(d[r
126                         --], sm);
127                 }
128                 return r + 1 - size;
129             }
130             sm = op(d[r], sm);
131             while((r & -r) != r);
132             return 0;
133         }
134     };

```

1.3 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 <<
6 1], st[v << 1 | 1]); }
7 segtree(int _n) : segtree(vector<S>(_n, e
8 ())) {}
9 segtree(const vector<S>& a) : n(sz(a)) {
10 log = __lg(2 * n - 1), size = 1 << log;
11 st.resize(size << 1, e());
12 REP(i, n) st[size + i] = a[i];
13 for(int i = size - 1; i; i--) update(i);
14 }
15 void set(int p, S val) {
16 st[p += size] = val;
17 for(int i = 1; i <= log; ++i) update(p
18 >> i);
19 }
20 S get(int p) const {
21 return st[p + size];
22 }
23 S prod(int l, int r) const {
24 assert(0 <= l && l <= r && r <= n);
25 S sm1 = e(), smr = e();
26 l += size, r += size;
27 while(l < r) {
28 if(l & 1) sm1 = op(sm1, st[l++]);
29 if(r & 1) smr = op(st[--r], smr);
30 l >>= 1;
31 r >>= 1;
32 }
33 return op(sm1, smr);
34 }
35 S all_prod() const { return st[1]; }
36 template<class F> int max_right(int l, F f
37 ) const {
38 assert(0 <= l && l <= n && f(e()));
39 if(l == n) return n;
40 l += size;
41 S sm = e();
42 do {
43 while(~l & 1) l >>= 1;
44 if(!f(op(sm, st[l]))) {
45 while(l < size) {
46 l <<= 1;
47 if(f(op(sm, st[l]))) sm = op(sm,
48 st[l++]);
49 }
50 return l - size;
51 }
52 sm = op(sm, st[l++]);
53 } while((l & -1) != 1);
54 return n;
55 }
56 template<class F> int min_left(int r, F f)
57 const {
58 assert(0 <= r && r <= n && f(e()));
59 if(r == 0) return 0;
60 r += size;
61 S sm = e();
62 do {
63 r--;
64 while(r > 1 && (r & 1)) r >>= 1;
65 if(!f(op(st[r], sm))) {
66 while(r < size) {
67 r <<= 1;
68 if(f(op(st[r], sm))) sm = op(st[r
69 --], sm);
70 }
71 return r + 1 - size;
72 }
73 sm = op(st[r], sm);
74 } while((r & -r) != r);
75 return 0;
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

1.4 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     {}
7     int lg = __lg(n) + 1;
8     b.resize(lg); b[0] = a;
9     for(int j = 1; j < lg; ++j) {
10         b[j].resize(n - (1 << j) + 1);
11         REP(i, n - (1 << j) + 1) b[j][i] = op(
12             b[j - 1][i], b[j - 1][i + (1 << j
13             - 1)]);
14     }
15     T prod(int from, int to) {
16         int lg = __lg(to - from + 1);
17         return op(b[lg][from], b[lg][to - (1 <<
18         lg) + 1]);
19     }
20 }
21 }
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

1.5 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     }
25     else {
26         b->l = merge(b->l, a);
27     }
28     pull(a), pull(b);
29     return a;
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

```

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         ;
41         if(p.second) p.second->p = NULL;
42         v->r = p.first;
43         pull(v); return {v, p.second};
44     }
45 }
46 int get_position(Node* v) { // 0-indexed
47     int k = (v->l != NULL ? v->l->sz : 0);
48     while(v->p != NULL) {
49         if(v == v->p->r) {
50             k++;
51             if(v->p->l != NULL) k += v->p->l->sz;
52         }
53         v = v->p;
54     }
55     return k;
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

1.6 動態凸包

```

1 struct line_t {
2     mutable ll k, m, p;
3     bool operator<(const line_t& o) const {
4         return k < o.k; }
5     bool operator<(ll x) const { return p < x;
6     }
7 };
8 template<bool MAX>
9 struct CHT : multiset<line_t, less<>> {
10     const ll INF = 1e18L;
11     bool isect(iterator x, iterator y) {
12         if(y == end()) return x->p = INF, 0;
13         if(x->k == y->k) {
14             x->p = (x->m > y->m ? INF : -INF);
15         } else {
16             x->p = floor_div(y->m - x->m, x->k - y
17             ->k); // see Math
18         }
19         return x->p >= y->p;
20     }
21 }
22 void add_line(ll k, ll m) {
23     if(!MAX) k = -k, m = -m;
24     auto z = insert({k, m, 0}), y = z++, x =
25     y;
26     while(isect(y, z)) z = erase(z);
27     if(x != begin() && isect(--x, y)) isect(
28     x, y = erase(y));
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

```

23 while((y = x) != begin() && (--x)->p >=
24 y->p) isect(x, erase(y));
25 }
26 ll get(ll x) {
27     assert(!empty());
28     auto l = *lower_bound(x);
29     return (1.k * x + 1.m) * (MAX ? +1 : -1)
30     ;
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

1.7 回滾 DSU

```

1 struct RollbackDSU {
2     int n; vi sz, tag;
3     vector<tuple<int, int, int>> op;
4     void init(int _n) {
5         n = _n;
6         sz.assign(n, -1);
7         tag.clear();
8     }
9     int leader(int x) {
10         while(sz[x] >= 0) x = sz[x];
11         return x;
12     }
13     bool merge(int x, int y) {
14         x = leader(x), y = leader(y);
15         if(x == y) return false;
16         if(-sz[x] < -sz[y]) swap(x, y);
17         op.pb(x, sz[x], y, sz[y]);
18         sz[x] += sz[y]; sz[y] = x;
19         return true;
20     }
21     int size(int x) { return -sz[leader(x)]; }
22     void add_tag() { tag.pb(sz[op]); }
23     void rollback() {
24         int z = tag.back(); tag.ppb();
25         while(sz[op] > z) {
26             auto [x, sx, y, sy] = op.back(); op.
27             ppb();
28             sz[x] = sx;
29             sz[y] = sy;
30         }
31     }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

2 Flow-Matching

2.1 Dinic

```

1 template<class T>
2 class Dinic {
3 public:
4     struct Edge {
5         int from, to;
6         T cap;
7         Edge(int x, int y, T z) : from(x), to(y)
8         , cap(z) {}
9     }
10 }
11 }
12 }
13 }
14 }
15 }
16 }
17 }
18 }
19 }
20 }
21 }
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

2.2 Flow 建模

- Maximum/Minimum flow with lower bound / Circulation problem

- Construct super source S and sink T .
- For each edge (x, y, l, u) , connect $x \rightarrow y$ with capacity $u - l$.
- For each vertex v , denote by $in(v)$ the difference between the sum of incoming lower bounds and the sum of outgoing lower bounds.
- If $in(v) > 0$, connect $S \rightarrow v$ with capacity $in(v)$, otherwise, connect $v \rightarrow T$ with capacity $-in(v)$.

- To maximize, connect $t \rightarrow s$ 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 $t \rightarrow s$ 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.

- 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)

- Redirect every edge: $y \rightarrow x$ if $(x, y) \in M$, $x \rightarrow y$ otherwise.
- DFS from unmatched vertices in X .
- $x \in X$ is chosen iff x is unvisited.
- $y \in Y$ is chosen iff y is visited.

- Minimum cost cyclic flow

- Construct super source S and sink T
- For each edge (x, y, c) , connect $x \rightarrow y$ with $(cost, cap) = (c, 1)$ if $c > 0$, otherwise connect $y \rightarrow x$ with $(cost, cap) = (-c, 1)$
- For each edge with $c < 0$, sum these cost as K , then increase $d(y)$ by 1, decrease $d(x)$ by 1
- For each vertex v with $d(v) > 0$, connect $S \rightarrow v$ with $(cost, cap) = (0, d(v))$
- For each vertex v with $d(v) < 0$, connect $v \rightarrow T$ with $(cost, cap) = (0, -d(v))$
- Flow from S to T , the answer is the cost of the flow $C + K$

- Maximum density induced subgraph

- Binary search on answer, suppose we're checking answer T
- Construct a max flow model, let K be the sum of all weights
- Connect source $s \rightarrow v, v \in G$ with capacity K
- For each edge (u, v, w) in G , connect $u \rightarrow v$ and $v \rightarrow u$ with capacity w
- For $v \in G$, connect it with sink $v \rightarrow t$ with capacity $K + 2T - (\sum_{e \in E(v)} w(e)) - 2w(v)$
- T is a valid answer if the maximum flow $f < K|V|$

- Minimum weight edge cover

- For each $v \in V$ create a copy v' , and connect $u' \rightarrow v'$ with weight $w(u, v)$.

- Connect $v \rightarrow v'$ with weight $2\mu(v)$, where $\mu(v)$ is the cost of the cheapest edge incident to v .
- Find the minimum weight perfect matching on G' .

- Project selection problem

- If $p_v > 0$, create edge (s, v) with capacity p_v ; otherwise, create edge (v, t) with capacity $-p_v$.
- Create edge (u, v) with capacity w with w being the cost of choosing u without choosing v .
- The mincut is equivalent to the maximum profit of a subset of projects.

- 0/1 quadratic programming

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

can be minimized by the mincut of the following graph:

- Create edge (x, t) with capacity c_x and create edge (s, y) with capacity c_y .
- Create edge (x, y) with capacity c_{xy} .
- Create edge (x, y) and edge (x', y') with capacity $c_{xyx'y'}$.

2.3 KM

```
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;
}
};
```

2.4 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.pb(u, v, cap, cost);
        g[v].pb(sz(edges));
        edges.pb(v, u, 0, -cost);
    }
    bool spfa(int s, int t) {
        bool found = false;
        fill(all(d), INF);
        d[s] = 0;

```

```

28 inq[s] = true;
29 queue<int> q;
30 q.push(s);
31 while(!q.empty()) {
32     int u = q.front(); q.pop();
33     if(u == t) found = true;
34     inq[u] = false;
35     for(auto& id : g[u]) {
36         const auto& e = edges[id];
37         if(e.cap > 0 && d[u] + e.cost < d[e.to]) {
38             d[e.to] = d[u] + e.cost;
39             pedge[e.to] = id;
40             if(!inq[e.to]) {
41                 q.push(e.to);
42                 inq[e.to] = true;
43             }
44         }
45     }
46     return found;
47 }
48 pair<S, T> flow(int s, int t, S f = INF) {
49     S cap = 0;
50     T cost = 0;
51     while(f > 0 && spfa(s, t)) {
52         S send = f;
53         int u = t;
54         while(u != s) {
55             const Edge& e = edges[pedge[u]];
56             send = min(send, e.cap);
57             u = e.from;
58         }
59         u = t;
60         while(u != s) {
61             Edge& e = edges[pedge[u]];
62             e.cap -= send;
63             Edge& b = edges[pedge[u] ^ 1];
64             b.cap += send;
65             u = e.from;
66         }
67         cap += send;
68         f -= send;
69         cost += send * d[t];
70     }
71     return {cap, cost};
72 }
73 }
74 };

```

2.5 一般圖最大匹配

```

1 struct GeneralMaxMatch {
2     int n;
3     vector<pii> es;
4     vi g, vis, mate; // i 與 mate[i] 配對 (
5     // mate[i] == -1 代表沒有匹配)
6     GeneralMaxMatch(int n) : n(n), g(n, -1),
7     mate(n, -1) {}
8     bool dfs(int u) {
9         if(vis[u]) return false;
10        vis[u] = true;
11        for(int ei = g[u]; ei != -1; ) {

```

```

10        auto [x, y] = es[ei]; ei = y;
11        if(mate[x] == -1) {
12            mate[mate[u] = x] = u;
13            return true;
14        }
15        for(int ei = g[u]; ei != -1; ) {
16            auto [x, y] = es[ei]; ei = y;
17            int nu = mate[x];
18            mate[mate[u] = x] = u;
19            mate[nu] = -1;
20            if(dfs(nu)) return true;
21            mate[mate[nu] = x] = nu;
22            mate[u] = -1;
23        }
24        return false;
25    }
26    void add_edge(int a, int b) {
27        auto f = [&](int a, int b) {
28            es.pb(b, g[a]);
29            g[a] = sz(es) - 1;
30        };
31        f(a, b); f(b, a);
32    }
33    int solve() {
34        vi o(n);
35        iota(all(o), 0);
36        int ans = 0;
37        REP(it, 100) {
38            shuffle(all(o), rng);
39            vis.assign(n, false);
40            for(auto i : o) if(mate[i] == -1) ans += dfs(i);
41        }
42        return ans;
43    }
44 }
45 };

```

2.6 一般圖最小權完美匹配

```

1 struct Graph {
2     // Minimum General Weighted Matching (
3     // Perfect Match) 0-base
4     static const int MXN = 105;
5     int n, edge[MXN][MXN];
6     int match[MXN], dis[MXN], onstk[MXN];
7     vector<int> stk;
8     void init(int _n) {
9         n = _n;
10        for(int i=0; i<n; i++)
11            for(int j=0; j<n; j++)
12                edge[i][j] = 0;
13    }
14    void add_edge(int u, int v, int w) { edge[
15        u][v] = edge[v][u] = w; }
16    bool SPFA(int u) {
17        if(onstk[u]) return true;
18        stk.push_back(u);
19        onstk[u] = 1;
20        for(int v=0; v<n; v++) {
21            if(u != v && match[u] != v && !onstk[v]) {
22                int m = match[v];

```

```

21        if(dis[m] > dis[u] - edge[v][m] +
22            edge[u][v]){
23            dis[m] = dis[u] - edge[v][m] +
24                edge[u][v];
25            onstk[v] = 1;
26            stk.push_back(v);
27            if(SPFA(m)) return true;
28            stk.pop_back();
29            onstk[v] = 0;
30        }
31        onstk[u] = 0;
32        stk.pop_back();
33        return false;
34    }
35    int solve() {
36        for(int i = 0; i < n; i += 2) match[i] =
37            i + 1, match[i+1] = i;
38        while(true) {
39            int found = 0;
40            for(int i=0; i<n; i++) dis[i] = onstk[
41                i] = 0;
42            for(int i=0; i<n; i++){
43                stk.clear();
44                if(!onstk[i] && SPFA(i)){
45                    found = 1;
46                    while(stk.size()>2){
47                        int u = stk.back(); stk.pop_back
48                            ();
49                        int v = stk.back(); stk.pop_back
50                            ();
51                        match[u] = v;
52                        match[v] = u;
53                    }
54                }
55                if(!found) break;
56            }
57            int ans = 0;
58            for(int i=0; i<n; i++) ans += edge[i][
59                match[i]];
60            return ans / 2;
61        }
62    }
63 }graph;

```

2.7 二分圖最大匹配

```

1 struct bipartite_matching {
2     int n, m; // 二分圖左右人數 (0 ~ n-1), (0
3     ~ m-1)
4     vector<vi> g;
5     vi lhs, rhs, dist; // i 與 lhs[i] 配對 (
6     // lhs[i] == -1 代表沒有配對)
7     bipartite_matching(int _n, int _m) : n(_n),
8     m(_m), g(_n), lhs(_n, -1), rhs(_m,
9     -1), dist(_n) {}
10    void add_edge(int u, int v) { g[u].pb(v);
11    }
12    void bfs() {
13        queue<int> q;
14        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]] ==
21                -1) {
22                dist[rhs[v]] = dist[u] + 1;
23                q.push(rhs[v]);
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(
36                rhs[v])) {
37                rhs[lhs[u] = v] = u;
38                return true;
39            }
40        }
41        return false;
42    }
43    int solve() {
44        int ans = 0;
45        while(true) {
46            bfs();
47            int aug = 0;
48            REP(i, n) if(lhs[i] == -1) aug += dfs(
49                i);
50            if(!aug) break;
51            ans += aug;
52        }
53    }
54    return ans;
55 }
56 };

```

3 Geometry

3.1 convex-hull

```

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(
5         dots))) {
6         for(int i = 1, t = sz(ans); i < sz(dots)
7             ; ans.pb(dots[i++])) {
8             while(sz(ans) > t && ori(ans[sz(ans) -
9                 2], ans.back(), dots[i]) < 0) {

```

```

7     ans.ppb();
8 }
9 }
10 }
11 ans.ppb();
12 swap(ans, dots);
13 }

```

3.2 point-in-convex-hull

```

1 int point_in_convex_hull(const vector<P>& a,
2   P p) {
3     // -1 ON, 0 OUT, +1 IN
4     // 要先逆時針排序
5     int n = sz(a);
6     if(btw(a[0], a[1], p) || btw(a[0], a[n -
7       1], p)) return -1;
8     int l = 0, r = n - 1;
9     while(l <= r) {
10        int m = (l + r) / 2;
11        auto a1 = cross(a[m] - a[0], p - a[0]);
12        auto a2 = cross(a[(m + 1) % n] - a[0], p
13          - a[0]);
14        if(a1 >= 0 && a2 <= 0) {
15            auto res = cross(a[(m + 1) % n] - a[m
16              ], p - a[m]);
17            return res > 0 ? 1 : (res >= 0 ? -1 :
18              0);
19        }
20        if(a1 < 0) r = m - 1;
21        else l = m + 1;
22    }
23    return 0;
24 }

```

3.3 point

```

1 using P = pair<ll, ll>;
2
3 P operator+(P a, P b) { return P{a.X + b.X,
4   a.Y + b.Y}; }
5 P operator-(P a, P b) { return P{a.X - b.X,
6   a.Y - b.Y}; }
7 P operator*(P a, ll b) { return P{a.X * b, a
8   .Y * b}; }
9 P operator/(P a, ll b) { return P{a.X / b, a
10   .Y / b}; }
11 ll dot(P a, P b) { return a.X * b.X + a.Y *
12   b.Y; }
13 ll cross(P a, P b) { return a.X * b.Y - a.Y
14   * b.X; }
15 ll abs2(P a) { return dot(a, a); }
16 double abs(P a) { return sqrt(abs2(a)); }
17 int sign(ll x) { return x < 0 ? -1 : (x == 0
18   ? 0 : 1); }
19 int ori(P a, P b, P c) { return sign(cross(b
20   - a, c - a)); }
21 bool collinear(P a, P b, P c) { return sign(
22   cross(a - c, b - c)) == 0; }

```

```

14 bool btw(P a, P b, P c) {
15     if(!collinear(a, b, c)) return 0;
16     return sign(dot(a - c, b - c)) <= 0;
17 }
18 bool seg_intersect(P a, P b, P c, P d) {
19     int a123 = ori(a, b, c);
20     int a124 = ori(a, b, d);
21     int a341 = ori(c, d, a);
22     int a342 = ori(c, d, b);
23     if(a123 == 0 && a124 == 0) {
24         return btw(a, b, c) || btw(a, b, d) ||
25           btw(c, d, a) || btw(c, d, b);
26     }
27     return a123 * a124 <= 0 && a341 * a342 <=
28       0;
29 }
30 P intersect(P a, P b, P c, P d) {
31     int a123 = cross(b - a, c - a);
32     int a124 = cross(b - a, d - a);
33     return (d * a123 - c * a124) / (a123 -
34       a124);
35 }

```

3.4 polar-angle-sort

```

1 bool cmp(P a, P b) {
2     #define ng(k) (sign(k.Y) < 0 || (sign(k.Y)
3       == 0 && sign(k.X) < 0))
4     int A = ng(a), B = ng(b);
5     if(A != B) return A < B;
6     if(sign(cross(a, b)) == 0) return abs2(a)
7       < abs2(b);
8     return sign(cross(a, b)) > 0;
9 }

```

3.5 最近點對

```

1 const ll INF = 9e18L;
2 vector<P> a;
3 sort(all(a), [](P a, P b) { return a.x < b.x
4   ; });
5 ll solve(int l, int r) {
6     if(l + 1 == r) return INF;
7     int m = (l + r) / 2;
8     ll d = min(solve(l, m), solve(m, r));
9     inplace_merge(a.begin() + l, a.begin() + m
10       , a.begin() + r, [](P a, P b) {
11         return a.y < b.y;
12       });
13     #define SQ(x) ((x) * (x))
14     vector<P> p;
15     for(int i = l; i < r; ++i) if(SQ(a[i].x -
16       a[m].x) <= d) p.pb(a[i]);
17     REP(i, sz(p)) {
18         for(int j = i + 1; j < sz(p); ++j) {
19             if((p[i].y - p[j].y) * (p[i].y - p[j].
20               y) > d) break;
21             d = min(d, SQ(p[i].x - p[j].x) + SQ(p[
22               i].y - p[j].y));
23         }
24     }
25     return d;
26 }

```

```

18     }
19 }
20 return d; // 距離平方
21 }

```

4 Graph

4.1 2-SAT

```

1 struct two_sat {
2     int n; SCC g;
3     vector<bool> ans;
4     two_sat(int _n) : n(_n), g(_n * 2) {}
5     void add_or(int u, bool x, int v, bool y) {
6         g.add_edge(2 * u + !x, 2 * v + y);
7         g.add_edge(2 * v + !y, 2 * u + x);
8     }
9     bool solve() {
10        ans.resize(n);
11        auto id = g.solve();
12        REP(i, n) {
13            if(id[2 * i] == id[2 * i + 1]) return
14              false;
15            ans[i] = (id[2 * i] < id[2 * i + 1]);
16        }
17        return true;
18    }
19 }

```

4.2 centroid-tree

```

1 pair<int, vector<vi>> centroid_tree(const
2   vector<vi>& g) {
3     int n = sz(g);
4     vi siz(n);
5     vector<bool> vis(n);
6     auto dfs_sz = [&](auto f, int u, int p) ->
7       void {
8         siz[u] = 1;
9         for(auto v : g[u]) {
10            if(v == p || vis[v]) continue;
11            f(f, v, u);
12            siz[u] += siz[v];
13        }
14        auto find_cd = [&](auto f, int u, int p,
15          int all) -> int {
16            for(auto v : g[u]) {
17                if(v == p || vis[v]) continue;
18                if(siz[v] * 2 > all) return f(f, v, u,
19                  all);
20            }
21            return u;
22        };
23        vector<vi> h(n);
24        auto build = [&](auto f, int u) -> int {
25            dfs_sz(dfs_sz, u, -1);
26        };
27    };
28 }

```

```

23     int cd = find_cd(find_cd, u, -1, siz[u])
24       ;
25     vis[cd] = true;
26     for(auto v : g[cd]) {
27         if(vis[v]) continue;
28         int child = f(f, v);
29         h[cd].pb(child);
30     }
31     return cd;
32 };
33 int root = build(build, 0);
34 return {root, h};
35 }

```

4.3 HLD

```

1 struct HLD {
2     int n;
3     vector<vi> g;
4     vi siz, par, depth, top, tour, fi, id;
5     sparse_table<pii, min> st;
6     HLD(int _n) : n(_n), g(_n), siz(_n), par(
7       _n), depth(_n), top(_n), fi(_n), id(_n
8       ) {}
9     tour.reserve(n);
10    void add_edge(int u, int v) {
11        g[u].push_back(v);
12        g[v].push_back(u);
13    }
14    void build(int root = 0) {
15        par[root] = -1;
16        top[root] = root;
17        vector<pii> euler_tour;
18        euler_tour.reserve(2 * n - 1);
19        dfs_sz(root);
20        dfs_link(euler_tour, root);
21        st = sparse_table<pii, min>(euler_tour);
22    }
23    int get_lca(int u, int v) {
24        int L = fi[u], R = fi[v];
25        if(L > R) swap(L, R);
26        return st.prod(L, R).second;
27    }
28    bool is_anc(int u, int v) {
29        return id[u] <= id[v] && id[v] < id[u] +
30          siz[u];
31    }
32    bool on_path(int a, int b, int x) {
33        return (is_ancestor(x, a) || is_ancestor
34          (x, b)) && is_ancestor(get_lca(a, b),
35            x);
36    }
37    int get_dist(int u, int v) {
38        return depth[u] + depth[v] - 2 * depth[
39          get_lca(u, v)];
40    }
41    int kth_anc(int u, int k) {
42        if(depth[u] < k) return -1;
43        int d = depth[u] - k;
44        while(depth[top[u]] > d) u = par[top[u]
45          ];
46        return tour[id[u] + d - depth[u]];
47    }
48 }

```



```

41 }
42 int kth_node_on_path(int a, int b, int k)
43 {
44     int z = get_lca(a, b);
45     int fi = depth[a] - depth[z];
46     int se = depth[b] - depth[z];
47     if(k < 0 || k > fi + se) return -1;
48     if(k < fi) return kth_anc(a, k);
49     return kth_anc(b, fi + se - k);
50 }
51 vector<pii> get_path(int u, int v, bool
52     include_lca = true) {
53     if(u == v && !include_lca) return {};
54     vector<pii> seg;
55     while(top[u] != top[v]) {
56         if(depth[top[u]] > depth[top[v]]) swap
57             (u, v);
58         seg.eb(id[top[v]], id[v]);
59         v = par[top[v]];
60     }
61     if(depth[u] > depth[v]) swap(u, v); // u
62     is lca
63     if(u != v || include_lca) seg.eb(id[u] +
64         !include_lca, id[v]);
65     return seg;
66 }
67 void dfs_sz(int u) {
68     if(par[u] != -1) g[u].erase(find(all(g[u
69         ]), par[u]));
70     siz[u] = 1;
71     for(auto& v : g[u]) {
72         par[v] = u;
73         depth[v] = depth[u] + 1;
74         dfs_sz(v);
75         siz[u] += siz[v];
76         if(siz[v] > siz[g[u][0]]) swap(v, g[u
77             ][0]);
78     }
79 }
80 void dfs_link(vector<pii>& euler_tour, int
81     u) {
82     fi[u] = sz(euler_tour);
83     id[u] = sz(tour);
84     euler_tour.eb(depth[u], u);
85     tour.pb(u);
86     for(auto v : g[u]) {
87         top[v] = (v == g[u][0] ? top[u] : v);
88         dfs_link(euler_tour, v);
89         euler_tour.eb(depth[u], u);
90     }
91 }

```

4.4 lowlink

```

1 struct lowlink {
2     int n, cnt = 0, tecc_cnt = 0, tvcc_cnt =
3         0;
4     vector<vector<pii>> g;
5     vector<pii> edges;
6     vi roots, id, low, tecc_id, tvcc_id;
7     vector<bool> is_bridge, is_cut,
8         is_tree_edge;

```

```

9     lowlink(int _n : n(_n), g(_n), is_cut(_n,
10         false), id(_n, -1), low(_n, -1) {}
11 void add_edge(int u, int v) {
12     g[u].eb(v, sz(edges));
13     g[v].eb(u, sz(edges));
14     edges.eb(u, v);
15     is_bridge.pb(false);
16     is_tree_edge.pb(false);
17     tvcc_id.pb(-1);
18 }
19 void dfs(int u, int peid = -1) {
20     static vi stk;
21     static int rid;
22     if(peid < 0) rid = cnt;
23     if(peid == -1) roots.pb(u);
24     id[u] = low[u] = cnt++;
25     for(auto [v, eid] : g[u]) {
26         if(eid == peid) continue;
27         if(id[v] < id[u]) stk.pb(eid);
28         if(id[v] >= 0) {
29             low[u] = min(low[u], id[v]);
30             if(id[u] == rid && id[v] != rid +
31                 1 || (id[u] != rid && low[v] >=
32                     id[u])) {
33                 is_cut[u] = true;
34             }
35             if(low[v] >= id[u]) {
36                 while(true) {
37                     int e = stk.back();
38                     stk.pop_back();
39                     tvcc_id[e] = tvcc_cnt;
40                     if(e == eid) break;
41                 }
42                 tvcc_cnt++;
43             }
44         }
45     }
46 void build() {
47     REP(i, n) if(id[i] < 0) dfs(i);
48     REP(i, sz(edges)) {
49         auto [u, v] = edges[i];
50         if(id[u] > id[v]) swap(u, v);
51         is_bridge[i] = (id[u] < low[v]);
52     }
53 }
54 vector<vi> two_ecc() { // 邊雙
55     tecc_cnt = 0;
56     tecc_id.assign(n, -1);
57     vi stk;
58     REP(i, n) {
59         if(tecc_id[i] != -1) continue;
60         tecc_id[i] = tecc_cnt;
61         stk.pb(i);
62         while(sz(stk)) {
63             int u = stk.back(); stk.pop_back();
64             for(auto [v, eid] : g[u]) {
65                 if(tecc_id[v] >= 0 || is_bridge[
66                     eid]) {
67                     continue;

```

```

68         tecc_id[v] = tecc_cnt;
69         stk.pb(v);
70     }
71     tecc_cnt++;
72 }
73 vector<vi> comp(tecc_cnt);
74 REP(i, n) comp[tecc_id[i]].pb(i);
75 return comp;
76 }
77 vector<vi> bcc_vertices() { // 點雙
78     vector<vi> comp(tvcc_cnt);
79     REP(i, sz(edges)) {
80         comp[tvcc_id[i]].pb(edges[i].first);
81         comp[tvcc_id[i]].pb(edges[i].second);
82     }
83     for(auto& v : comp) {
84         sort(all(v));
85         v.erase(unique(all(v)), v.end());
86     }
87     REP(i, n) if(g[i].empty()) comp.pb({i});
88     return comp;
89 }
90 vector<vi> bcc_edges() {
91     vector<vi> ret(tvcc_cnt);
92     REP(i, sz(edges)) ret[tvcc_id[i]].pb(i);
93     return ret;
94 }
95 }
96 };

```

4.5 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
21             u, int x) {
22             id[u] = x;
23             for(auto v : h[u]) {
24                 if(id[v] == -1) {
25                     dfs2(v, x);
26                 }
27             }
28         };
29         for(int i = n - 1, cnt = 0; i >= 0; --i)
30             {

```

```

29         int u = top[i];
30         if(id[u] == -1) {
31             dfs2(u, cnt);
32             cnt += 1;
33         }
34     }
35     return id;
36 }
37 };

```

5 Math

5.1 Chinese-Remainder

```

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

```

5.2 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)
6         != b) A[1LL * e * b % m] = j++;
7     if(e == b) return j;
8     if(__gcd(m, e) == __gcd(m, b)) {
9         for(int i = 2; i < n + 2; ++i) {
10             e = 1LL * e * f % m;
11             if(A.find(e) != A.end()) return n * i
12                 - A[e];
13         }
14     }
15     return -1;

```

5.3 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 }

```

5.4 Floor-Sum

```

1 // sum_{i=0}^{n-1} floor((a * i + b) / m) in Log
  (n + m + a + b)
2 ll floor_sum(ll n, ll m, ll a, ll b) {
3     ll ans = 0;
4     if(a >= m) ans += (n - 1) * n * (a / m) /
5         2, a %= m;
6     if(b >= m) ans += n * (b / m), b %= m;
7     ll y_max = (a * n + b) / m, x_max = (y_max
8         * m - b);
9     if(y_max == 0) return ans;
10    ans += (n - (x_max + a - 1) / a) * y_max;
11    return ans + floor_sum(y_max, a, m, (a -
12        x_max % a) % a);
13 }

```

5.5 Miller-Rabin

```

1 bool is_prime(ll n, vector<ll> x) {
2     ll d = n - 1;
3     d >= __builtin_ctzll(d);
4     for(auto a : x) {
5         if(n <= a) break;
6         ll t = d, y = 1, b = t;
7         while(b) {
8             if(b & 1) y = i128(y) * a % n;
9             a = i128(a) * a % n;
10            b >>= 1;
11        }
12        while(t != n - 1 && y != 1 && y != n -
13            1) {
14            y = i128(y) * y % n;
15            t <<= 1;
16        }
17        if(y != n - 1 && t % 2 == 0) return
18            false;
19    }
20    return true;
21 }
22 bool is_prime(ll n) {
23     if(n <= 1) return false;
24     if(n % 2 == 0) return n == 2;
25     if(n < (1LL << 30)) return is_prime(n, {2,
26         7, 61});
27     return is_prime(n, {2, 325, 9375, 28178,
28         450775, 9780504, 1795265022});
29 }

```

5.6 Pollard-Rho

```

1 void PollardRho(map<ll, int>& mp, ll n) {
2     if(n == 1) return;
3     if(is_prime(n)) return mp[n]++, void();
4     if(n % 2 == 0) {
5         mp[2] += 1;
6         PollardRho(mp, n / 2);
7         return;
8     }
9     ll x = 2, y = 2, d = 1, p = 1;
10    #define f(x, n, p) ((i128(x) * x % n + p)
11        % n)
12    while(true) {
13        if(d != 1 && d != n) {
14            PollardRho(mp, d);
15            PollardRho(mp, n / d);
16            return;
17        }
18        p += (d == n);
19        x = f(x, n, p), y = f(f(y, n, p), n, p);
20        d = __gcd(abs(x - y), n);
21    }
22    #undef f
23 }
24 vector<ll> get_divisors(ll n) {
25     if(n == 0) return {};
26     map<ll, int> mp;
27     PollardRho(mp, n);
28     vector<pair<ll, int>> v(all(mp));
29     vector<ll> res;
30     auto f = [&](auto f, int i, ll x) -> void
31         {
32             if(i == sz(v)) {
33                 res.pb(x);
34                 return;
35             }
36             for(int j = v[i].second; ; j--) {
37                 f(f, i + 1, x);
38                 if(j == 0) break;
39                 x *= v[i].first;
40             }
41             f(f, 0, 1);
42             sort(all(res));
43             return res;
44         }

```

5.7 Primes

```

1 /* 12721 13331 14341 75577 123457 222557
   556679 999983 1097774749 1076767633
   100102021 999997771 1001010013
   1000512343 987654361 999991231 999888733
   98789101 98777773 999991921 1010101333
   1010102101 1000000000039
   1000000000000037 2305843009213693951
   4611686018427387847 9223372036854775783
   18446744073709551557 */

```

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

5.9 定理

• Cramer's rule

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

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

• 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 $d_{ij} = -d_{ji}$. $\frac{\text{rank}(D)}{2}$ is the maximum matching on G .

• Cayley's Formula

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

• Erdős–Gallai theorem

A sequence of nonnegative integers $d_1 \geq \dots \geq d_n$ can be represented as the degree sequence of a finite simple graph on n vertices if and only if $d_1 + \dots + d_n$ 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 $a_1 \geq \dots \geq a_n$ and b_1, \dots, b_n is bigraphic if and only if $\sum_{i=1}^n a_i = \sum_{i=1}^n b_i$ and $\sum_{i=1}^k a_i \leq \sum_{i=1}^k \min(b_i, k)$ holds for every $1 \leq k \leq n$.

• Fulkerson–Chen–Anstee theorem

A sequence $(a_1, b_1), \dots, (a_n, b_n)$ of nonnegative integer pairs with $a_1 \geq \dots \geq a_n$ 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

$$\begin{aligned} f(n) &= \sum_{d|n} g(d) \Leftrightarrow g(n) = \sum_{d|n} \mu(d) f\left(\frac{n}{d}\right) \\ f(n) &= \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu\left(\frac{d}{n}\right) f(d) \end{aligned}$$

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

5.10 整數除法

```

1 ll floor_div(ll a, ll b) {
2     return a/b - ((a^b) < 0 && a%b != 0);
3 }
4 ll ceil_div(ll a, ll b) {
5     return a/b + ((a^b) > 0 && a%b != 0);
6 }

```

5.11 數字

- Bernoulli numbers

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

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

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

- 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. $\pi(j) > j$.

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

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

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

5.12 歐幾里得類算法

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

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

$$g(a, b, c, n) = \sum_{i=0}^n i \lfloor \frac{ai+b}{c} \rfloor = \begin{cases} \lfloor \frac{a}{c} \rfloor \cdot \frac{n(n+1)(2n+1)}{6} + \lfloor \frac{b}{c} \rfloor \cdot \frac{n(n+1)}{2} + g(a \bmod c, b \bmod c, c, n), & a \geq c \vee b \geq c \\ 0, & n < 0 \vee a < 0 \vee b < 0 \\ \frac{1}{2} \cdot (n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1)), & \text{otherwise} \end{cases}$$

$$h(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor^2 = \begin{cases} \lfloor \frac{a}{c} \rfloor^2 \cdot \frac{n(n+1)(2n+1)}{6} + \lfloor \frac{b}{c} \rfloor^2 \cdot (n+1) + \lfloor \frac{a}{c} \rfloor \cdot \lfloor \frac{b}{c} \rfloor \cdot n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2\lfloor \frac{a}{c} \rfloor \cdot g(a \bmod c, b \bmod c, c, n) + 2\lfloor \frac{b}{c} \rfloor \cdot f(a \bmod c, b \bmod c, c, n), & a \geq c \vee b \geq c \\ 0, & n < 0 \vee a < 0 \vee b < 0 \\ nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a, b, c, n), & \text{otherwise} \end{cases}$$

5.13 生成函數

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

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

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

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

- Special Generating Function

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

6 Misc

6.1 default

```
#define all(v) (v).begin(), (v).end()
#define sz(v) ((int)v.size())
#define pb push_back
#define eb emplace_back
#define pbb pop_back
#define REP(i, n) for(int i = 0; i < int(n); i++)

using vi = vector<int>;
using pii = pair<int, int>;
a ≥ c ∨ b ≥ c
n < 0 ∨ a = 0
```

6.2 PBDS

```
#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));
a ≥ c ∨ b ≥ c
n < 0 ∨ a = 0
```

6.3 python

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

6.4 readchar

```
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++;
}
```

6.5 矩形覆蓋面積

```
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(1 < 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[j][0] == ev[i][0])
            update(ev[j][1], ev[j][2] - 1, ev[j][3]);
        j++;
        prev = ev[i][0], i = j - 1;
    }
    cout << ans << "\n";
}
```

7 String

7.1 AC

7.2 KMP

```
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;
    }
}
```



```

9 | }
10 | return k;
11 | }

```

7.3 LCP

```

1 | vector<int> lcp(const vector<int>& s, const
  | vector<int>& sa) {
2 |     int n = sz(s), h = 0;
3 |     vi rnk(n), lcp(n - 1);
4 |     REP(i, n) rnk[sa[i]] = i;
5 |     REP(i, n) {
6 |         h -= (h > 0);
7 |         if(rnk[i] == 0) continue;
8 |         int j = sa[rnk[i] - 1];
9 |         for(; j + h < n && i + h < n; h++) if(s[
10 |             j + h] != s[i + h]) break;
11 |         lcp[rnk[i] - 1] = h;
12 |     }
13 |     return lcp;

```

7.4 manacher

```

1 | // Length: (z[i] - (i & 1)) / 2 * 2 + (i &
  | 1)
2 | vi manacher(string t) {
3 |     string s = "&";
4 |     for(char c : t) s.pb(c), s.pb('%');
5 |     int l = 0, r = 0;
6 |     vi z(sz(s));
7 |     REP(i, sz(s)) {
8 |         z[i] = r > i ? min(z[2 * l - i], r - i)
9 |             : 1;
10 |         while(s[i + z[i]] == s[i - z[i]]) z[i]
11 |             ++;
12 |         if(z[i] + i > r) r = z[i] + 1, l = i;
13 |     }
14 |     return z;

```

7.5 rolling-hash

```

1 | const ll M = 911382323, mod = 972663749;
2 | ll Get(vector<ll>& v, int l, int r) {
3 |     if(!l) return h[r]; // p[i] = M^i % mod
4 |     ll ans = (h[r] - h[l - 1] * p[r - l + 1])
5 |         % mod;
6 |     return (ans + mod) % mod;
7 | }
8 | vector<ll> Hash(string s) {
9 |     vector<ll> ans(sz(s));
10 |     ans[0] = s[0];
11 |     for(int i = 1; i < sz(s); i++) ans[i] = (
12 |         ans[i - 1] * M + s[i]) % mod;
13 |     return ans;

```

7.6 SAIS

```

1 | // mississippi
2 | // 10 7 4 1 0 9 8 6 3 5 2
3 | vi SAIS(string a) {
4 |     #define QQ(i, n) for(int i = (n); i >= 0;
5 |         i--)
6 |     int n = sz(a), m = *max_element(all(a)) +
7 |         1;
8 |     vi pos(m + 1), x(m), sa(n), val(n), lms;
9 |     for(auto c : a) pos[c + 1]++;
10 |     REP(i, m) pos[i + 1] += pos[i];
11 |     vector<bool> s(n);
12 |     QQ(i, n - 2) s[i] = a[i] != a[i + 1] ? a[i]
13 |         < a[i + 1] : s[i + 1];
14 |     auto ind = [&](const vi& ls){
15 |         fill(all(sa), -1);
16 |         auto L = [&](int i) { if(i >= 0 && !s[i]
17 |             ) sa[x[a[i]]++] = i; };
18 |         auto S = [&](int i) { if(i >= 0 && s[i])
19 |             sa[--x[a[i]]] = i; };
20 |         REP(i, m) x[i] = pos[i + 1];
21 |         QQ(i, sz(ls) - 1) S(ls[i]);
22 |         REP(i, m) x[i] = pos[i];
23 |         L(n - 1);
24 |         REP(i, n) L(sa[i] - 1);
25 |         REP(i, m) x[i] = pos[i + 1];
26 |         QQ(i, n - 1) S(sa[i] - 1);
27 |     };
28 |     auto ok = [&](int i) { return i == n || (!
29 |         s[i - 1] && s[i]); };
30 |     auto same = [&](int i, int j) {
31 |         do {
32 |             if(a[i++] != a[j++]) return false;
33 |             while(!ok(i) && !ok(j));
34 |             return ok(i) && ok(j);
35 |         };
36 |         for(int i = 1; i < n; i++) if(ok(i)) lms.
37 |             pb(i);
38 |         ind(lms);
39 |         if(sz(lms)) {
40 |             int p = -1, w = 0;
41 |             for(auto v : sa) if(v && ok(v)) {
42 |                 if(p != -1 && same(p, v)) w--;
43 |                 val[p = v] = w++;
44 |             }
45 |             auto b = lms;
46 |             for(auto& v : b) v = val[v];
47 |             b = SAIS(b);
48 |             for(auto& v : b) v = lms[v];
49 |             ind(b);
50 |         }
51 |     }
52 |     return sa;

```

7.7 SAM

```

1 | // cnt 要先用 bfs 往回推, 第一次出現的位置是
  | state.first_pos - |S| + 1
2 | struct Node { int go[26], len, link, cnt,
  |     first_pos; };
3 | Node SA[N]; int sz;

```

```

4 | void sa_init() { SA[0].link = -1, SA[0].len
  |     = 0, sz = 1; }
5 | int sa_extend(int p, int c) {
6 |     int u = sz++;
7 |     SA[u].first_pos = SA[u].len = SA[p].len +
8 |         1;
9 |     SA[u].cnt = 1;
10 |     while(p != -1 && SA[p].go[c] == 0) {
11 |         SA[p].go[c] = u;
12 |         p = SA[p].link;
13 |     }
14 |     if(p == -1) {
15 |         SA[u].link = 0;
16 |         return u;
17 |     }
18 |     int q = SA[p].go[c];
19 |     if(SA[p].len + 1 == SA[q].len) {
20 |         SA[u].link = q;
21 |         return u;
22 |     }
23 |     int x = sz++;
24 |     SA[x] = SA[q];
25 |     SA[x].cnt = 0;
26 |     SA[x].len = SA[p].len + 1;
27 |     SA[q].link = SA[u].link = x;
28 |     while(p != -1 && SA[p].go[c] == q) {
29 |         SA[p].go[c] = x;
30 |         p = SA[p].link;
31 |     }
32 |     return u;

```

7.8 smallest-rotation

```

1 | string small_rot(string s) {
2 |     int n = sz(s), i = 0, j = 1;
3 |     s += s;
4 |     while(i < n && j < n) {
5 |         int k = 0;
6 |         while(k < n && s[i + k] == s[j + k]) k
7 |             ++;
8 |         if(s[i + k] <= s[j + k]) j += k + 1;
9 |         else i += k + 1;
10 |         if(i == j) j++;
11 |     }
12 |     int ans = i < n ? i : j;
13 |     return s.substr(ans, n);

```

7.9 Z

```

1 | vi z_algorithm(const vi& a) {
2 |     int n = sz(a);
3 |     vi z(n);
4 |     for(int i = 1, j = 0; i < n; ++i) {
5 |         if(i <= j + z[j]) z[i] = min(z[i - j], j
6 |             + z[j] - i);
7 |         while(i + z[i] < n && a[i + z[i]] == a[
8 |             z[i]]) z[i]++;
9 |         if(i + z[i] > j + z[j]) j = i;

```

ACM ICPC Team Reference - Angry Crow Takes Flight!

Contents

1 Data-Structure	1
1.1 fast-set	1
1.2 lazysegtree	1
1.3 segtree	1
1.4 sparse-table	2

1.5 treap	2
1.6 動態凸包	2
1.7 回滾 DSU	2
2 Flow-Matching	2
2.1 Dinic	2
2.2 Flow 建模	3
2.3 KM	3
2.4 MCMF	3
2.5 一般圖最大匹配	4
2.6 一般圖最小權完美匹配	4
2.7 二分圖最大匹配	4
3 Geometry	4
3.1 convex-hull	4
3.2 point-in-convex-hull	5
3.3 point	5
3.4 polar-angle-sort	5
3.5 最近點對	5

4 Graph	5
4.1 2-SAT	5
4.2 centroid-tree	5
4.3 HLD	5
4.4 lowlink	6
4.5 SCC	6
5 Math	6
5.1 Chinese-Remainder	6
5.2 Discrete-Log	6
5.3 extgcd	6
5.4 Floor-Sum	7
5.5 Miller-Rabin	7
5.6 Pollard-Rho	7
5.7 Primes	7
5.8 估計值	7
5.9 定理	7
5.10 整數除法	7
5.11 數字	8

5.12 歐幾里得類算法	8
5.13 生成函數	8
6 Misc	8
6.1 default	8
6.2 PBDS	8
6.3 python	8
6.4 readchar	8
6.5 矩形覆蓋面積	8
7 String	8
7.1 AC	8
7.2 KMP	8
7.3 LCP	9
7.4 manacher	9
7.5 rolling-hash	9
7.6 SAIS	9
7.7 SAM	9
7.8 smallest-rotation	9
7.9 Z	9

ACM ICPC Judge Test - Angry Crow Takes Flight!

C++ Resource Test

```
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
14                             // block_size 改成常數
15     memset(ptr, 'a', block_size);
16     cout << depth << endl;
17     stack_size_dfs(depth + 1);
18 }
19
20 void stack_size_and_runtime_error(size_t
21     block_size, size_t bound = 1024) {
22     system_test::block_size = block_size;
23     system_test::bound = bound;
24     stack_size_dfs();
25 }
26
27 double speed(int iter_num) {
28     const int block_size = 1024;
29     volatile int A[block_size];
30     auto begin = chrono::high_resolution_clock
31         ::now();
32     while (iter_num--)
33         for (int j = 0; j < block_size; ++j)
34             A[j] += j;
35     auto end = chrono::high_resolution_clock::
36         now();
```

```
37 chrono::duration<double> diff = end -
38     begin;
39     return diff.count();
40 }
41
42 void runtime_error_1() {
43     // Segmentation fault
44     int *ptr = nullptr;
45     *(ptr + 7122) = 7122;
46 }
47
48 void runtime_error_2() {
49     // Segmentation fault
50     int *ptr = (int *)memset;
51     *ptr = 7122;
52 }
53
54 void runtime_error_3() {
55     // munmap_chunk(): invalid pointer
56     int *ptr = (int *)memset;
57     delete ptr;
58 }
59
60 void runtime_error_4() {
61     // free(): invalid pointer
62     int *ptr = new int[7122];
63     ptr += 1;
64     delete[] ptr;
65 }
```

```
62
63 void runtime_error_5() {
64     // maybe illegal instruction
65     int a = 7122, b = 0;
66     cout << (a / b) << endl;
67 }
68
69 void runtime_error_6() {
70     // floating point exception
71     volatile int a = 7122, b = 0;
72     cout << (a / b) << endl;
73 }
74
75 void runtime_error_7() {
76     // call to abort.
77     assert(false);
78 }
79
80 } // namespace system_test
81
82 #include <sys/resource.h>
83 void print_stack_limit() { // only work in
84     Linux
85     struct rlimit l;
86     getrlimit(RLIMIT_STACK, &l);
87     cout << "stack_size = " << l.rlim_cur << "
88         byte" << endl;
89 }
```