

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     {
6         int sz = n * m;
7         first = siz = L = R = U = D = col = row
8         = stk = vi(sz);
9         REP(i, m + 1) {
10             L[i] = i - 1, R[i] = i + 1;
11             U[i] = D[i] = i;
12         }
13         L[0] = m, R[m] = 0;
14     }
15     void insert(int r, int c) {
16         r++, c++;
17         col[++tot] = c, row[tot] = r, ++siz[c];
18         D[tot] = D[c], U[D[c]] = tot, U[tot] = c
19         , D[c] = tot;
20         if(!first[r]) first[r] = L[tot] = R[tot]
21         = tot;
22         else {
23             L[R[tot]] = R[first[r]] = tot;
24             R[L[tot]] = first[r] = tot;
25         }
26     }
27     #define TRAV(i, X, j) for(i = X[j]; i != j
28     ; i = X[i])
29     void remove(int c) {
30         int i, j;
31         L[R[c]] = L[c], R[L[c]] = R[c];
32         TRAV(i, D, c) TRAV(j, R, i) {
33             D[U[D[j]]] = U[j] = D[j];
34             siz[col[j]]--;
35         }
36     }
37     void recover(int c) {
38         int i, j;
39         TRAV(i, U, c) TRAV(j, L, i) {
40             U[D[j]] = D[U[j]] = j;
41             siz[col[j]]++;
42         }
43         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() +
57     ans);
58 }
```

2.2 fast-set

```
1 // Can correctly work with numbers in range
2 // Supports all std::set operations in O(1)
3 // on random queries / dense arrays, O(
4 // Log64(N)) in worst case (sparse array).
5 // Count operation works in O(1) always.
6 template<uint MAXN>
7 class fast_set {
8 private:
9     static const uint PREF = (MAXN <= 64 ? 0 :
10     MAXN <= 4096 ? 1 :
11     MAXN <= 262144 ? 1 + 64 :
12     MAXN <= 16777216 ? 1 + 64 +
13     4096 :
14     MAXN <= 1073741824 ? 1 + 64
15     + 4096 + 262144 : 227) +
16     1;
17     static constexpr ull lb(int x) {
18         if(x == 64) return ULLONG_MAX;
19         return (1ULL << x) - 1;
20     };
21     static const uint SZ = PREF + (MAXN + 63)
22     / 64 + 1;
23     ull m[SZ] = {0};
24     inline uint left(uint v) const { return (v
25     - 62) * 64; }
26     inline uint parent(uint v) const { return
27     v / 64 + 62; }
28     inline void setbit(uint v) { m[v >> 6] |=
29     1ULL << (v & 63); }
30     inline void resetbit(uint v) { m[v >> 6]
31     &= ~(1ULL << (v & 63)); }
32     inline uint getbit(uint v) const { return
33     m[v >> 6] >> (v & 63) & 1; }
34     inline ull childs_value(uint v) const {
35     return m[left(v) >> 6]; }
36     inline int left_go(uint x, const uint c)
37     const {
38         const ull rem = x & 63;
```

```
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 =
30     parent(bt)) {
31     const ull rem = bt & 63;
32     num = m[bt >> 6] & lb(rem);
33     if(num) {
34         bt = (bt ^ rem) | __lg(num);
35         break;
36     }
37 }
38 if(bt == 62) return -1;
39 while(bt < PREF * 64) bt = left(bt) |
40     __lg(m[bt - 62]);
41 return bt - PREF * 64;
42 }
43 inline int right_go(uint x, const uint c)
44 const {
45     const ull rem = x & 63;
46     uint bt = PREF * 64 + x;
47     ull num = m[bt >> 6] & ~lb(rem + c);
48     if(num) return (x ^ rem) |
49     __builtin_ctzll(num);
50     for(bt = parent(bt); bt > 62; bt =
51     parent(bt)) {
52     const ull rem = bt & 63;
53     num = m[bt >> 6] & ~lb(rem + 1);
54     if(num) {
55         bt = (bt ^ rem) | __builtin_ctzll(
56         num);
57         break;
58     }
59 }
60 if(bt == 62) return -1;
61 while(bt < PREF * 64) bt = left(bt) |
62     __builtin_ctzll(m[bt - 62]);
63 return bt - PREF * 64;
64 }
65 public:
66 fast_set() { assert(PREF != 228); setbit
67 (62); }
68 bool empty() const {return getbit(63);}
69 void clear() { fill(m, m + SZ, 0); setbit
70 (62); }
71 bool count(uint x) const { return m[PREF +
72 (x >> 6)] >> (x & 63) & 1; }
73 void insert(uint x) { for(uint v = PREF *
74 64 + x; !getbit(v); v = parent(v))
75     setbit(v); }
76 void erase(uint x) {
77     if(!getbit(PREF * 64 + x)) return;
78     resetbit(PREF * 64 + x);
79     for(uint v = parent(PREF * 64 + x); v >
80     62 && !childs_value(v); v = parent(v))
81     resetbit(v);
82 }
83 int find_next(uint x) const { return
84     right_go(x, 0); } // >=
85 int find_prev(uint x) const { return
86     left_go(x, 1); } // <=
87 }
```

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],
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    }
18    void push(int k) {
19        all_apply(k << 1, lz[k]);
20        all_apply(k << 1 | 1, lz[k]);
21        lz[k] = id();
22    }
23    lazy_segtree(int _n) : lazy_segtree(vector
24    <S>(_n, e())) {}
25    lazy_segtree(const vector<S>& v) : n(sz(v))
26    {
27        log = __lg(2 * n - 1), size = 1 << log;
28        d.resize(size * 2, e());
29        lz.resize(size, id());
30        REP(i, n) d[size + i] = v[i];
31        for(int i = size - 1; i; i--) update(i);
32    }
33    void set(int p, S x) {
34        p += size;
35        for(int i = log; i; --i) push(p >> i);
36        d[p] = x;
37        for(int i = 1; i <= log; ++i) update(p
38        >> i);
39    }
40    S get(int p) {
41        p += size;
42        for(int i = log; i; i--) push(p >> i);
43        return d[p];
44    }
45    S prod(int l, int r) {
46        if(l == r) return e();
47        l += size; r += size;
48        for(int i = log; i; i--) {
49            if(((l >> i) << i) != 1) push(l >> i);
50            if(((r >> i) << i) != r) push(r >> i);
51        }
52        S sm1 = e(), smr = e();
53        while(l < r) {
54            if(l & 1) sm1 = op(sm1, d[l++]);
55            if(r & 1) smr = op(d[--r], smr);
56            l >>= 1; r >>= 1;
57        }
58        return op(sm1, smr);
59    }
60    S all_prod() const { return d[1]; }
61    void apply(int p, F f) {
62        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) != 1) 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) != 1) 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) != 1);
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) != 1);
49    return n;
50 }
51 template<class F> int min_left(int r, F f)
    const {
52     assert(0 <= r && r <= n && f(e()));
53     if(r == 0) return 0;
54     r += size;
55     S sm = e();
56     do {
57         r--;
58         while(r > 1 && (r & 1)) r >>= 1;
59         if(!f(op(st[r], sm))) {
60             while(r < size) {
61                 r = r << 1 | 1;
62                 if(f(op(st[r], sm))) sm = op(st[r]
                    --, sm);
63             }
64             return r + 1 - size;
65         }
66         sm = op(st[r], sm);
67     } while((r & -r) != r);
68     return 0;
69 }
70 };

```

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 ? 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 {
4         return k < o.k; }
5     bool operator<(ll x) const { return p < x;
6     }
7 };
8 template<bool MAX>

```

```

7 struct CHT : multiset<line_t, less<>> {
8     const ll INF = 1e18L;
9     bool isect(iterator x, iterator y) {
10         if(y == end()) return x->p = INF, 0;
11         if(x->k == y->k) {
12             x->p = (x->m > y->m ? INF : -INF);
13         } else {
14             x->p = floor_div(y->m - x->m, x->k - y->k); // see Math
15         }
16         return x->p >= y->p;
17     }
18     void add_line(ll k, ll m) {
19         if(!MAX) k = -k, m = -m;
20         auto z = insert({k, m, 0}), y = z++, x = y;
21         while(isect(y, z)) z = erase(z);
22         if(x != begin() && isect(--x, y)) isect(x, y = erase(y));
23         while((y = x) != begin() && (--x)->p >= y->p) isect(x, erase(y));
24     }
25     ll get(ll x) {
26         assert(!empty());
27         auto l = *lower_bound(x);
28         return (l.k * x + l.m) * (MAX ? +1 : -1);
29     }
30 };

```

2.8 回滾 DSU

```

1 struct RollbackDSU {
2     int n; vi sz, tag;
3     vector<tuple<int, 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.ppb();
27             sz[x] = sx;
28             sz[y] = sy;
29         }
30     }

```

```
31 |};
```

3 Flow-Matching

3.1 Dinic-LowerBound

```

1 template<class T>
2 struct DinicLowerBound {
3     using Maxflow = Dinic<T>;
4     int n;
5     Maxflow d;
6     vector<T> in;
7     DinicLowerBound(int _n) : n(_n), d(_n + 2)
8         , in(_n) {}
9     int add_edge(int from, int to, T low, T
10         high) {
11         assert(0 <= low && low <= high);
12         in[from] -= low, in[to] += low;
13         return d.add_edge(from, to, high - low);
14     }
15     T flow(int s, int t) {
16         T sum = 0;
17         REP(i, n) {
18             if(in[i] > 0) {
19                 d.add_edge(n, i, in[i]);
20                 sum += in[i];
21             }
22             if(in[i] < 0) d.add_edge(i, n + 1, -in[i]);
23         }
24         d.add_edge(t, s, numeric_limits<T>::max());
25         if(d.flow(n, n + 1) < sum) return -1;
26         return d.flow(s, t);
27     }
28 };

```

3.2 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), cap(z) {}
8     };
9     constexpr T INF = 1e9;
10    int n;
11    vector<Edge> edges;
12    vector<vi> g;
13    vi cur, h; // h : Level graph
14    Dinic(int _n) : n(_n), g(_n) {}
15    void add_edge(int u, int v, T c) {
16        g[u].pb(sz(edges));
17        edges.pb({u, v, c});
18        g[v].pb(sz(edges));

```

```

19        edges.pb({v, u, 0});
20    }
21    bool bfs(int s, int t) {
22        h.assign(n, -1);
23        queue<int> q;
24        h[s] = 0;
25        q.push(s);
26        while(!q.empty()) {
27            int u = q.front(); q.pop();
28            for(int i : g[u]) {
29                const auto& e = edges[i];
30                int v = e.to;
31                if(e.cap > 0 && h[v] == -1) {
32                    h[v] = h[u] + 1;
33                    if(v == t) return true;
34                    q.push(v);
35                }
36            }
37        }
38        return false;
39    }
40    T dfs(int u, int t, T f) {
41        if(u == t) return f;
42        T r = f;
43        for(int& i = cur[u]; i < sz(g[u]); ++i) {
44            int j = g[u][i];
45            const auto& e = edges[j];
46            int v = e.to;
47            T c = e.cap;
48            if(c > 0 && h[v] == h[u] + 1) {
49                T a = dfs(v, t, min(r, c));
50                edges[j].cap -= a;
51                edges[j ^ 1].cap += a;
52                if((r -= a) == 0) return f;
53            }
54        }
55        return f - r;
56    }
57    T flow(int s, int t, T f = INF) {
58        T ans = 0;
59        while(f > 0 && bfs(s, t)) {
60            cur.assign(n, 0);
61            T cur = dfs(s, t, f);
62            ans += cur;
63            f -= cur;
64        }
65        return ans;
66    }
67 };

```

3.3 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

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

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

```
1 template<class T>
2 struct KM {
3     static constexpr T INF = numeric_limits<T>
4         >::max();
5     int n, ql, qr;
6     vector<vector<T>> w;
7     vector<T> hl, hr, slk;
8     vi fl, fr, pre, qu;
9     vector<bool> vl, vr;
10    KM(int n) : n(n), w(n, vector<T>(n, -INF)),
11        hl(n), hr(n), slk(n), fl(n), fr(n),
12        pre(n), qu(n), vl(n), vr(n) {}
13    void add_edge(int u, int v, int x) { w[u][
14        v] = x; } // 最小值要加負號
15    bool check(int x) {
16        vl[x] = 1;
17        if(fl[x] != -1) return vr[qu[qr++]] = fl[
18            x] = 1;
19        while(x != -1) swap(x, fr[fl[x] = pre[x
20            ]]);
21        return 0;
22    }
23    void bfs(int s) {
24        fill(all(slk), INF);
25        fill(all(vl), 0);
26        fill(all(vr), 0);
27        ql = qr = 0, qu[qr++] = s, vr[s] = 1;
28        while(true) {
29            T d;
30            while(ql < qr) {
31                for(int x = 0, y = qu[ql++]; x < n;
32                    ++x) {
33                    if(!vl[x] && slk[x] >= (d = hl[x]
34                        + hr[y] - w[x][y])) {
35                        pre[x] = y;
36                        if(d) slk[x] = d;
37                        else if(!check(x)) return;
38                    }
39                }
40            }
41            d = INF;
42            REP(x, n) if(!vl[x] && d > slk[x]) d =
43                slk[x];
44            REP(x, n) {
45                if(vl[x]) hl[x] += d;
46                else slk[x] -= d;
47                if(vr[x]) hr[x] -= d;
48            }
49        }
50    }
```

```
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

```
1 template<class S, class T>
2 class MCMF {
3 public:
4     struct Edge {
5         int from, to;
6         S cap;
7         T cost;
8         Edge(int u, int v, S x, T y) : from(u),
9             to(v), cap(x), cost(y) {}
10    };
11    const ll INF = 1e18L;
12    int n;
13    vector<Edge> edges;
14    vector<vi> g;
15    vector<T> d;
16    vector<bool> inq;
17    vi pedge;
18    MCMF(int _n) : n(_n), g(_n), d(_n), inq(_n
19        ), pedge(_n) {}
20    void add_edge(int u, int v, S cap, T cost) {
21        g[u].pb(sz(edges));
22        edges.pb({u, v, cap, cost});
23        g[v].pb(sz(edges));
24        edges.pb({v, u, 0, -cost});
25    }
26    bool spfa(int s, int t) {
27        bool found = false;
28        fill(all(d), INF);
29        d[s] = 0;
30        inq[s] = true;
31        queue<int> q;
32        q.push(s);
33        while(!q.empty()) {
34            int u = q.front(); q.pop();
35            if(u == t) found = true;
36            inq[u] = false;
37            for(auto& id : g[u]) {
38                const auto& e = edges[id];
39                if(e.cap > 0 && d[u] + e.cost < d[e.
40                    to]) {
41                    d[e.to] = d[u] + e.cost;
42                    pedge[e.to] = id;
43                }
44            }
45        }
46        return found;
47    }
```

```
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

```
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; ) {
12            auto [x, y] = es[ei]; ei = y;
13            if(mate[x] == -1) {
14                mate[mate[u] = x] = u;
15                return true;
16            }
17        }
18        for(int ei = g[u]; ei != -1; ) {
19            auto [x, y] = es[ei]; ei = y;
20            int nu = mate[x];
21            mate[mate[u] = x] = nu;
22            mate[nu] = -1;
23            if(dfs(nu)) return true;
24            mate[mate[nu] = x] = nu;
25        }
26        return false;
27    }
```

```
mate[u] = -1;
}
return false;
}
void add_edge(int a, int b) {
    auto f = [&](int a, int b) {
        es.pb({a, b});
        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

```
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                ]){
23                int m = match[v];
24                if(dis[m] > dis[u] - edge[v][m] +
25                    edge[u][v]){
26                    dis[m] = dis[u] - edge[v][m] +
27                        edge[u][v];
28                    onstk[v] = 1;
29                    stk.push_back(v);
30                    if(SPFA(m)) return true;
31                    onstk[v] = 0;
32                }
33            }
34        }
35        onstk[u] = 0;
36    }
```

```

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++){
41             dis[i] = onstk[
42                 i] = 0;
43             for(int i=0; i<n; i++){
44                 stk.clear();
45                 if(!onstk[i] && SPFA(i)){
46                     found = 1;
47                     while(stk.size())>=2){
48                         int u = stk.back(); stk.pop_back
49                             ();
50                         int v = stk.back(); stk.pop_back
51                             ();
52                         match[u] = v;
53                         match[v] = u;
54                     }
55                     if(!found) break;
56                 }
57             }
58             int ans = 0;
59             for(int i=0; i<n; i++) ans += edge[i][
60                 match[i]];
61             return ans / 2;
62         }
63     }
64 }
65 }graph;

```

3.8 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) {
15            if(lhs[i] == -1) {
16                q.push(i);
17                dist[i] = 0;
18            } else {
19                dist[i] = -1;
20            }
21        }
22        while(!q.empty()) {
23            int u = q.front(); q.pop();
24            for(auto v : g[u]) {
25                if(rhs[v] != -1 && dist[rhs[v]] ==
26                    -1) {
27                    dist[rhs[v]] = dist[u] + 1;

```

```

22         q.push(rhs[v]);
23     }
24 }
25 }
26 bool dfs(int u) {
27     for(auto v : g[u]) {
28         if(rhs[v] == -1) {
29             rhs[lhs[u] = v] = u;
30             return true;
31         }
32     }
33     for(auto v : g[u]) {
34         if(dist[rhs[v]] == dist[u] + 1 && dfs(
35             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(
48             i);
49         if(!aug) break;
50         ans += aug;
51     }
52     return ans;
53 }

```

4 Geometry

4.1 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) {
10                 ans.ppb();
11             }
12         }
13     }
14     ans.ppb();
15     swap(ans, dots);
16 }

```

4.2 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 }

```

4.3 point

```

1 using P = pair<ll, ll>;
2 P operator+(P a, P b) { return P{a.X + b.X,
3     a.Y + b.Y}; }
4 P operator-(P a, P b) { return P{a.X - b.X,
5     a.Y - b.Y}; }
6 P operator*(P a, ll b) { return P{a.X * b, a
7     .Y * b}; }
8 P operator/(P a, ll b) { return P{a.X / b, a
9     .Y / b}; }
10 ll dot(P a, P b) { return a.X * b.X + a.Y *
11     b.Y; }
12 ll cross(P a, P b) { return a.X * b.Y - a.Y
13     * b.X; }
14 ll abs2(P a) { return dot(a, a); }
15 double abs(P a) { return sqrt(abs2(a)); }
16 int sign(ll x) { return x < 0 ? -1 : (x == 0
17     ? 0 : 1); }
18 int ori(P a, P b, P c) { return sign(cross(b
19     - a, c - a)); }
20 bool collinear(P a, P b, P c) { return sign(
21     cross(a - c, b - c)) == 0; }
22 bool btw(P a, P b, P c) {
23     if(!collinear(a, b, c)) return 0;
24     return sign(dot(a - c, b - c)) <= 0;
25 }
26 bool seg_intersect(P a, P b, P c, P d) {
27     int a123 = ori(a, b, c);
28     int a124 = ori(a, b, d);
29     int a341 = ori(c, d, a);
30     int a342 = ori(c, d, b);
31     if(a123 == 0 && a124 == 0) {
32         return btw(a, b, c) || btw(a, b, d) ||
33             btw(c, d, a) || btw(c, d, b);
34     }
35     return a123 * a124 <= 0 && a341 * a342 <=
36         0;
37 }

```

```

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 -
32         a124);
33 }
34 struct line { P A, B; };
35 P vec(line L) { return L.B - L.A; }
36 P projection(P p, line L) { return L.A + vec
37     (L) / abs(vec(L)) * dot(p - L.A, vec(L))
38     / abs(vec(L)); }

```

4.4 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 }

```

4.5 最近點對

```

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

```


5 Graph

5.1 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     {
7         g.add_edge(2 * u + !x, 2 * v + y);
8         g.add_edge(2 * v + !y, 2 * u + x);
9     }
10    bool solve() {
11        ans.resize(n);
12        auto id = g.solve();
13        REP(i, n) {
14            if(id[2 * i] == id[2 * i + 1]) return
15                false;
16            ans[i] = (id[2 * i] < id[2 * i + 1]);
17        }
18    };

```

5.2 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     };
15     auto find_cd = [&](auto f, int u, int p,
16         int all) -> int {
17         for(auto v : g[u]) {
18             if(v == p || vis[v]) continue;
19             if(siz[v] * 2 > all) return f(f, v, u,
20                 all);
21         }
22         return u;
23     };
24     vector<vi> h(n);
25     auto build = [&](auto f, int u) -> int {
26         dfs_sz(dfs_sz, u, -1);
27         int cd = find_cd(find_cd, u, -1, siz[u]);
28         vis[cd] = true;
29         for(auto v : g[cd]) {
30             if(vis[v]) continue;
31             int child = f(f, v, u);
32             h[cd].pb(child);
33         }
34     };

```

```

29     }
30     return cd;
31 };
32 int root = build(build, 0);
33 return {root, h};
34 }

```

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

```

```

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

```

5.4 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].pb(v, sz(edges));
13         g[v].pb(u, sz(edges));
14         edges.pb(u, v);
15         is_bridge.pb(false);

```

```

13     is_tree_edge.pb(false);
14     tvcc_id.pb(-1);
15 }
16 void dfs(int u, int peid = -1) {
17     static vi stk;
18     static int rid;
19     if(peid < 0) rid = cnt;
20     if(peid == -1) roots.pb(u);
21     id[u] = low[u] = cnt++;
22     for(auto [v, eid] : g[u]) {
23         if(eid == peid) continue;
24         if(id[v] < id[u]) stk.pb(eid);
25         if(id[v] >= 0) {
26             low[u] = min(low[u], id[v]);
27         } else {
28             is_tree_edge[eid] = true;
29             dfs(v, eid);
30             low[u] = min(low[u], low[v]);
31             if((id[u] == rid && id[v] != rid +
32                 1) || (id[u] != rid && low[v] >=
33                 id[u])) {
34                 is_cut[u] = true;
35             }
36             if(low[v] >= id[u]) {
37                 while(true) {
38                     int e = stk.back();
39                     stk.pop_back();
40                     tvcc_id[e] = tvcc_cnt;
41                     if(e == eid) break;
42                 }
43                 tvcc_cnt++;
44             }
45         }
46     }
47     void build() {
48         REP(i, n) if(id[i] < 0) dfs(i);
49         REP(i, sz(edges)) {
50             auto [u, v] = edges[i];
51             if(id[u] > id[v]) swap(u, v);
52             is_bridge[i] = (id[u] < low[v]);
53         }
54     }
55     vector<vi> two_ecc() { // 邊雙
56         tecc_cnt = 0;
57         tecc_id.assign(n, -1);
58         vi stk;
59         REP(i, n) {
60             if(tecc_id[i] != -1) continue;
61             tecc_id[i] = tecc_cnt;
62             stk.pb(i);
63             while(sz(stk)) {
64                 int u = stk.back(); stk.pop_back();
65                 for(auto [v, eid] : g[u]) {
66                     if(tecc_id[v] >= 0 || is_bridge[
67                         eid]) {
68                         continue;
69                     }
70                     tecc_id[v] = tecc_cnt;
71                     stk.pb(v);
72                 }
73             }
74             tecc_cnt++;
75         }
76     }
77     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[tecc_id[i]].pb(edges[i].first);
82         comp[tecc_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[tecc_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
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             for(int i = n - 1, cnt = 0; i >= 0; --i)
29                 {
30                     int u = top[i];
31                     if(id[u] == -1) {
32                         dfs2(u, cnt);
33                         cnt += 1;
34                     }
35                 }
36             return id;

```

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
7             ? +1 : -1);
8         if(op < k) r = m;
9         else l = m + 1;
10    }
11    return f(l).first + l * k * (MAX ? +1 :

```

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,
4     998244352]) (mod 998244353)
5 // - [] -> (0, [1])
6 // - [0, 0, 0] -> (0, [1])
7 // - [-2] -> (1, [1, 2])
8 template<class T>
9 pair<int, vector<T>> BM(const vector<T>& S)
10 {
11     using poly = vector<T>;
12     int N = SZ(S);
13     poly C_rev{1}, B{1};
14     int L = 0, m = 1;
15     T b = 1;
16     auto adjust = [](poly C, const poly &B, T
17         d, T b, int m) -> poly {
18         C.resize(max(SZ(C), SZ(B) + m));
19         T a = d / b;
20         REP(i, SZ(B)) C[i + m] -= a * B[i];
21         return C;
22     };
23     REP(n, N) {
24         T d = S[n];
25         REP(i, L) d += C_rev[i + 1] * S[n - 1 -
26             i];
27         if(d == 0) m++;
28         else if (2 * L <= n) {
29             poly Q = C_rev;
30             C_rev = adjust(C_rev, B, d, b, m);
31             L = n + 1 - L, B = Q, b = d, m = 1;
32         } else C_rev = adjust(C_rev, B, d, b, m
33             ++);
34     }
35     return {L, C_rev};

```

```

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

```

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 {
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};

```

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)
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;

```

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

```
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 }
```

6.7 FWHT

```
1 #define ppc __builtin_popcount
2 template<class T, class F>
3 void fwht(vector<T>& a, F f) {
4     int n = SZ(a);
5     assert(ppc(n) == 1);
6     for(int i = 1; i < n; i <= 1) {
7         for(int j = 0; j < n; j += i < 1) {
8             REP(k, i) f(a[j + k], a[i + j + k]);
9         }
10    }
11 }
12 template<class T>
13 void or_transform(vector<T>& a, bool inv) {
14     fwht(a, [&](T& x, T& y) { y += x * (inv
15         ? -1 : +1); });
16 }
17 template<class T>
18 void and_transform(vector<T>& a, bool inv) {
19     fwht(a, [&](T& x, T& y) { x += y * (inv
20         ? -1 : +1); });
21 }
22 template<class T>
23 void xor_transform(vector<T>& a, bool inv) {
24     fwht(a, [&](T& x, T& y) { x += y * (inv
25         ? -1 : +1); });
26 }
27 }
28 template<class T>
29 vector<T> convolution(vector<T> a, vector<T>
30     b) {
31     assert(SZ(a) == SZ(b));
32     transform(a, false), transform(b, false);
33     REP(i, SZ(a)) a[i] *= b[i];
34     transform(a, true);
35     return a;
36 }
37 template<class T>
38 vector<T> subset_convolution(const vector<T>
39     & f, const vector<T> & g) {
```

```
38 assert(SZ(f) == SZ(g));
39 int n = SZ(f);
40 assert(ppc(n) == 1);
41 const int lg = __lg(n);
42 vector<vector<T>> fhat(lg + 1, vector<T>(n
43     )), ghat(fhat);
44 REP(i, n) fhat[ppc(i)][i] = f[i], ghat[ppc
45     (i)][i] = g[i];
46 REP(i, lg + 1) or_transform(fhat[i], false
47     ), or_transform(ghat[i], false);
48 vector<vector<T>> h(lg + 1, vector<T>(n));
49 REP(m, n) REP(i, lg + 1) REP(j, i + 1) h[i
50     ][m] += fhat[j][m] * ghat[i - j][m];
51 REP(i, lg + 1) or_transform(h[i], true);
52 vector<T> res(n);
53 REP(i, n) res[i] = h[ppc(i)][i];
54 return res;
55 }
```

6.8 Gauss-Jordan

```
1 int GaussJordan(vector<vector<ld>>& a) {
2     // -1 no sol, 0 inf sol
3     int n = SZ(a);
4     REP(i, n) assert(SZ(a[i]) == n + 1);
5     REP(i, n) {
6         int p = i;
7         REP(j, n) {
8             if(j < i && abs(a[j][j]) > EPS)
9                 continue;
10            if(abs(a[j][i]) > abs(a[p][i])) p = j;
11        }
12        REP(j, n + 1) swap(a[i][j], a[p][j]);
13        if(abs(a[i][i]) <= EPS) continue;
14        REP(j, n) {
15            if(i == j) continue;
16            ld delta = a[j][i] / a[i][i];
17            FOR(k, i, n + 1) a[j][k] -= delta * a
18                [i][k];
19        }
20    }
21    bool ok = true;
22    REP(i, n) {
23        if(abs(a[i][i]) <= EPS) {
24            if(abs(a[i][n]) > EPS) return -1;
25            ok = false;
26        }
27    }
28    return ok;
29 }
```

6.9 Linear-Sieve

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

```
8     phi[i] = i - 1;
9     mobius[i] = -1;
10 }
11 for(auto j : primes) {
12     if(i * j > n) break;
13     least[i * j] = j;
14     if(i % j == 0) {
15         mobius[i * j] = 0;
16         phi[i * j] = phi[i] * j;
17         break;
18     } else {
19         mobius[i * j] = -mobius[i];
20         phi[i * j] = phi[i] * phi[j];
21     }
22 }
23 }
24 }
```

```
8 if(n == -1) n = m - 1;
9 vi inv(n + 1);
10 inv[0] = inv[1] = 1;
11 for(int i = 2; i <= n; i++) inv[i] = m - 1
12     LL * (m / i) * inv[m % i] % m;
13 return inv;
```

6.12 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         };
42     sort(all(res));
43     return res;
44 }
```

6.10 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 }
```

6.11 Mod-Inv

```
1 int inv(int a) {
2     if(a < N) return inv[a];
3     if(a == 1) return 1;
4     return (MOD - 1LL * (MOD / a) * inv(MOD %
5         a) % MOD) % MOD;
6 }
7 vi mod_inverse(int m, int n = -1) {
8     assert(n < m);
```


6.13 Primes

```
1 /* 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: $B(n) = 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213507, 27644437, 190899322, \dots$

6.15 定理

• Cramer's rule

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

• 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, \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 $= \pi h^2 (3r - h)/3 = \pi h(3a^2 + h^2)/6 = \pi r^3 (2 + \cos \theta)/(3 \cos \theta)^2$.
- Area $= 2\pi r h = \pi(a^2 + h^2) = 2\pi r^2 (1 - \cos \theta)$.

6.16 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 }
```

6.17 數字

• Bernoulli numbers

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

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

$$S_m(n) = \sum_{k=1}^n k^m =$$

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

• 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. $\pi(j) > j$.

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

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

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

• 次方和

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

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

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

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

General form:

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

6.18 歐幾里得類算法

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

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

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

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

6.19 生成函數

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

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

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

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

• Special Generating Function

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

7 Misc

7.1 fast

```
1 #pragma GCC optimize("Ofast,no-stack-
   protector,unroll-loops,fast-math,inline
2 inline char gc() {
3     static const size_t sz = 65536;
4     static char buf[sz];
5     static char *p = buf, *end = buf;
6     if(p == end) end = buf + fread(buf, 1, sz,
7         stdin), p = buf;
8     return *p++;
9 }
```

7.2 next-combination

```
1 // Example: 1 -> 2, 4 -> 8, 12(1100) ->
   17(10001)
2 ll next_combination(ll comb) {
3     ll x = comb & -comb, y = comb + x;
4     return ((comb & ~y) / x >> 1) | y;
5 }
```

7.3 PBDS

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

7.4 python

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

7.5 rng

```
1 inline ull rng() {
2     static ull Q = 48763;
3     Q ^= Q << 7;
4     Q ^= Q >> 9;
5     return Q & 0xFFFFFFFFFULL;
6 }
```

7.6 rotate90

```
1 vector<vector<T>> rotate90(const vector<
   vector<T>>& a) {
2     int n = sz(a), m = sz(a[0]);
3     vector<vector<T>> b(m, vector<T>(n));
4     REP(i, n) REP(j, m) b[j][i] = a[i][m - 1
5         - j];
6     return b;
7 }
```

7.7 timer

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

7.8 矩形覆蓋面積

```
1 const int N = 2e6 + 5; // [-1e6, 1e6]
2 int tag[N * 4], seg[N * 4];
3 void pull(int v, int l, int r) {
4     seg[v] = 0;
5     if(tag[v] > 0) seg[v] = r - l + 1;
6     else if(1 < r) seg[v] = seg[v * 2] + seg[v
7         * 2 + 1];
8 }
9 void update(int ql, int qr, int x, int v =
10     1, int l = 0, int r = N - 1) {
11     if(ql > r || qr < l) return;
12     if(ql <= l && r <= qr) {
13         tag[v] += x;
14     } else {
15         int m = (l + r) / 2;
16         update(ql, qr, x, v * 2, l, m);
17         update(ql, qr, x, v * 2 + 1, m + 1, r);
18     }
19     pull(v, l, r);
20 }
21 int main() {
22     int n; cin >> n;
23     vector<array<int, 4>> ev(2 * n);
24     REP(i, n) {
25         int x, y, x2, y2;
26         cin >> x >> y >> x2 >> y2;
27         x += N / 2; y += N / 2;
28         x2 += N / 2; y2 += N / 2;
29         ev[2 * i] = {x, y, y2, +1};
30         ev[2 * i + 1] = {x2, y, y2, -1};
31     }
32     sort(all(ev));
33     ll ans = 0, prev = 0;
34     REP(i, 2 * n) {
35         ans += (ev[i][0] - prev) * seg[1];
36         int j = i;
37         while(j < 2 * n && ev[i][0] == ev[j][0])
38             update(ev[j][1], ev[j][2] - 1, ev[j]
39                 [3]);
40     }
```

```
37     j++;
38 }
39 prev = ev[i][0], i = j - 1;
40 }
41 cout << ans << "\n";
42 }
```

8 String

8.1 AC

```
1 template<int ALPHABET = 26, char MIN_CHAR =
   'a'>
2 struct ac_automaton {
3     struct Node {
4         int fail = 0, cnt = 0;
5         array<int, ALPHABET> go{};
6     };
7     vector<Node> node;
8     vi que;
9     int new_node() { return node.eb(), SZ(node)
10         - 1; }
11     Node& operator[](int i) { return node[i]; }
12     ac_automaton() { new_node(); }
13     int insert(const string& s) {
14         int p = 0;
15         for(char c : s) {
16             int v = c - MIN_CHAR;
17             if(node[p].go[v] == 0) node[p].go[v] =
18                 new_node();
19             p = node[p].go[v];
20         }
21         node[p].cnt++;
22         return p;
23     }
24     void build() {
25         que.reserve(SZ(node)); que.pb(0);
26         REP(i, SZ(que)) {
27             int u = que[i];
28             REP(j, ALPHABET) {
29                 if(node[u].go[j] == 0) node[u].go[j] =
30                     node[node[u].fail].go[j];
31                 else {
32                     int v = node[u].go[j];
33                     node[v].fail = (u == 0 ? u : node[
34                         u].fail).go[j];
35                     que.pb(v);
36                 }
37             }
38         }
39     }
```

8.2 KMP

```
1 // abacababa -> [0, 0, 1, 0, 0, 1, 2, 3]
2 vi KMP(const vi& a) {
```

```
3     int n = SZ(a);
4     vi k(n);
5     for(int i = 1; i < n; ++i) {
6         int j = k[i - 1];
7         while(j > 0 && a[i] != a[j]) j = k[j -
8             1];
9         j += (a[i] == a[j]);
10        k[i] = j;
11    }
12    return k;
13 }
```

8.3 LCP

```
1 vi lcp(const vi& s, const vi& 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;
14 }
```

8.4 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;
15 }
```

8.5 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 }

```

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

```

```

37 }
38 auto b = lms;
39 for(auto& v : b) v = val[v];
40 b = SAIS(b);
41 for(auto& v : b) v = lms[v];
42 ind(b);
43 }
44 return sa;
45 }

```

8.7 SAM

```

1 // cnt 要先用 bfs 往回推, 第一次出現的位置是
2 // state.first_pos - |S| + 1
3 struct Node { int go[26], len, link, cnt,
4     first_pos; };
5 Node SA[N]; int sz;
6 void sa_init() { SA[0].link = -1, SA[0].len
7     = 0, sz = 1; }
8 int sa_extend(int p, int c) {
9     int u = sz++;
10    SA[u].first_pos = SA[p].len = SA[p].len +
11        1;
12    SA[u].cnt = 1;
13    while(p != -1 && SA[p].go[c] == 0) {
14        SA[p].go[c] = u;
15        p = SA[p].link;
16    }
17    if(p == -1) {
18        SA[u].link = 0;
19        return u;
20    }
21    int q = SA[p].go[c];
22    if(SA[p].len + 1 == SA[q].len) {
23        SA[u].link = q;
24        return u;
25    }
26    int x = sz++;
27    SA[x] = SA[q];
28    SA[x].cnt = 0;
29    SA[x].len = SA[p].len + 1;
30    SA[q].link = SA[u].link = x;
31    while(p != -1 && SA[p].go[c] == q) {
32        SA[p].go[c] = x;
33        p = SA[p].link;
34    }
35    return u;
36 }

```

8.8 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    }
11 }

```

```

9     if(i == j) j++;
10 }
11 int ans = i < n ? i : j;
12 return s.substr(ans, n);
13 }

```

8.9 Z

```

1 // abacbababa -> [0, 0, 1, 0, 0, 3, 0, 1]
2 vi z_algorithm(const vi& a) {
3     int n = sz(a);
4     vi z(n);
5     for(int i = 1, j = 0; i < n; ++i) {
6         if(i <= j + z[j]) z[i] = min(z[i - j], j
7             + z[j] - i);
8         while(i + z[i] < n && a[i + z[i]] == a[z
9             [i]]) z[i]++;
10        if(i + z[i] > j + z[j]) j = i;
11    }
12    return z;
13 }

```

ACM ICPC Team Reference - NTHU SplayTreap

Contents

1 Basic	1	3 Flow-Matching	3	5.2 centroid-tree	6	6.18 歐幾里得類算法	9
1.1 vimrc	1	3.1 Dinic-LowerBound	3	5.3 HLD	6	6.19 生成函數	9
2 Data-Structure	1	3.2 Dinic	3	5.4 lowlink	6	7 Misc	10
2.1 DLX	1	3.3 Flow 建模	3	5.5 SCC	7	7.1 fast	10
2.2 fast-set	1	3.4 KM	4	6 Math	7	7.2 next-combination	10
2.3 lazysegtree	1	3.5 MCMF	4	6.1 Aliens	7	7.3 PBDS	10
2.4 segtree	2	3.6 一般圖最大匹配	4	6.2 Berlekamp-Massey	7	7.4 python	10
2.5 sparse-table	2	3.7 一般圖最小權完美匹配	4	6.3 Chinese-Remainder	7	7.5 rng	10
		3.8 二分圖最大匹配	5	6.4 Discrete-Log	7	7.6 rotate90	10
		4 Geometry	5	6.5 extgcd	7	7.7 timer	10
		4.1 convex-hull	5	6.6 Floor-Sum	8	7.8 矩形覆蓋面積	10
		4.2 point-in-convex-hull	5	6.7 FWHT	8	8 String	10
		4.3 point	5	6.8 Gauss-Jordan	8	8.1 AC	10
		4.4 polar-angle-sort	5	6.9 Linear-Sieve	8	8.2 KMP	10
		4.5 最近點對	5	6.10 Miller-Rabin	8	8.3 LCP	10
		5 Graph	6	6.11 Mod-Inv	8	8.4 manacher	10
		5.1 2-SAT	6	6.12 Pollard-Rho	8	8.5 rolling-hash	11
				6.13 Primes	9	8.6 SAIS	11
				6.14 估計值	9	8.7 SAM	11
				6.15 定理	9	8.8 smallest-rotation	11
				6.16 整數除法	9	8.9 Z	11
				6.17 數字	9		

ACM ICPC Judge Test - NTHU SplayTreap

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
10 size_t block_size, bound;
11 void stack_size_dfs(size_t depth = 1) {
12     if (depth >= bound)
13         return;
14     int8_t ptr[block_size]; // 若無法編譯將
15                             // block_size 改成常數
16     memset(ptr, 'a', block_size);
17     cout << depth << endl;
18     stack_size_dfs(depth + 1);
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 double speed(int iter_num) {
27     const int block_size = 1024;
28     volatile int A[block_size];
29     auto begin = chrono::high_resolution_clock
30         ::now();
31     while (iter_num--)
32         for (int j = 0; j < block_size; ++j)
33             A[j] += j;
34     auto end = chrono::high_resolution_clock::
35         now();
```

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

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