Contents

1 Environment

1.1 vimrc [fd479f]

```
se nu ai hls et ru ic is sc cul
se re=1 ts=4 sts=4 sw=4 ls=2 mouse=a
sy on
hi cursorline cterm=none ctermbg=89
ca hash w !cpp -dD -P -fpreprocessed \| tr -d
    '[:space:]' \| md5sum \| cut -c-6
```

1.2 vim grid paper script [85593f]

```
i|<esc>25A   |<esc>
o+<esc>25A---+<esc>
Vky35Pdd
```

1.3 build command [f8b6f9]

```
g++ code.cpp -DDEBUG -Wall -Wextra -fsanitize
   =address,undefined -O2 -std=c++17
```

1.4 stress test [a1a617]

```
for ((i = 1; ; ++i)); do
    ./gen > tc
    diff -w <(./code < tc) <(./hack < tc) ||
        break
done
```

2 Misc

2.1 random [48d1bc]

```
auto seed = chrono::steady_clock::now().
    time_since_epoch().count();
mt19937 rng(seed);

#include <ext/random>
__gnu_cxx::sfmt19937 rng(seed);

shuffle(begin, end, rng);
```

2.2 pbds [1c6a5a]

```
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;

gp_hash_table<K, V> hash_table;
cc_hash_table<K, V> hash_table;

tree<K, V/null_type, less<K>, rb_tree_tag,
     tree_order_statistics_node_update> t;
t.find_by_order(k);
t.order_of_key(val);
```

2.3 pragma [da629c]

```
#pragma GCC optimize("Ofast,unroll-loops")
#pragma GCC target("avx2,popcnt,lzcnt,abm,bmi
    ,bmi2,fma,tune=native")
```

2.4 debug template [fe3c64]

```
#ifdef DEBUG
#define debug(x) cerr << #x << " = " << (x)
    << endl;

template<typename T1, typename T2>
ostream &operator<<(ostream &o, const pair<T1
    , T2> &p);

template<typename T1, typename T2>
enable_if_t<is_same_v<T1, ostream>, T1 &>
    operator<<(T1 &o, const T2 &v) {
    o << '[';
    for (auto i = v.begin(); i != v.end(); ++
        i)
        o << (i != v.begin() ? " " : "") << *
            i;
    return o << ']';
}

template<typename T1, typename T2>
ostream &operator<<(ostream &o, const pair<T1
    , T2> &p) {
    return o << '(' << p.first << ':' << p.
        second << ')';
}
#else
#define debug(x) void()
#endif
```

2.5 int128 io [cebc20]

```
istream &operator>>(istream &is, __int128 &v)
    {
    string s;
    is >> s;
    v = 0;
    for (auto &&i: s)
        if (isdigit(i))
            v = v * 10 + i - '0';
    if (s[0] == '-') v *= -1;
    return is;
}

ostream &operator<<(ostream &os, const
    __int128 &v) {
    if (v == 0) return os << '0';
    __int128 num = v;
    if (v < 0) os << '-', num = -num;
    string s;
    while (num) s.push_back(num % 10 + '0'),
        num /= 10;
    reverse(s.begin(), s.end());
    return os << s;
}
```

2.6 modint [d12b20]

```
template<int M>
struct mint {
    int v;
    explicit operator int() const { return v;
        }

    mint(): v(0) {}

    mint(long long x): v(x % M) { v += (v <
        0) * M; }

    mint &operator+=(mint o) {
        if ((v += o.v) >= M) v -= M;
        return *this;
    }

    mint &operator-=(mint o) {
        if ((v -= o.v) < 0) v += M;
        return *this;
    }

    mint &operator*=(mint o) {
        v = 1ll * v * o.v % M;
        return *this;
    }

    mint &operator/=(mint o) {
        return *this *= inv(o);
    }
```

```
    friend mint pow(mint a, long long p) {
        assert(p >= 0);
        if (p >= M) p %= (M - 1);
        return p == 0 ? 1 : pow(a * a, p / 2)
            * (p & 1 ? a : 1);
    }

    friend mint inv(mint a) {
        assert(a.v != 0);
        return pow(a, M - 2);
    }

    friend int abs(mint a) { return a.v; }

    friend mint operator+(mint a, mint b) {
       return a += b; }
    friend mint operator-(mint a, mint b) {
       return a -= b; }
    friend mint operator*(mint a, mint b) {
       return a *= b; }
    friend mint operator/(mint a, mint b) {
       return a /= b; }
};
```

2.7 coordinate compression [57c822]

```
template<typename T>
vector<T> compress(vector<T> a) {
    sort(a.begin(), a.end());
    a.erase(unique(a.begin(), a.end()), a.end
       ());
    return a;
}

template<typename T>
int get_index(T val, const vector<T> &a) {
    return lower_bound(a.begin(), a.end(),
       val) - a.begin();
}
```

2.8 mo's algorithm [bec049]

```
struct query {
    int l, r, idx;
};

template<int K, typename T, T (*add)(int), T
   (*sub)(int)>
vector<T> mo(vector<query> q) {
    sort(q.begin(), q.end(), [](query &a,
       query &b) {
        if (a.l / K == b.l / K)
            return a.l / K & 1 ? a.r < b.r :
               a.r > b.r;
        return a.l < b.l;
    });
    T val;
    int l = 0, r = -1;
    vector<T> ans(q.size());
    for (auto &[ql, qr, idx]: q) {
        while (l < ql) val = sub(l++);
        while (l > ql) val = add(--l);
        while (r < qr) val = add(++r);
        while (r > qr) val = sub(r--);
        ans[idx] = val;
    }
    return ans;
}
```

2.9 two sat [bad513]

```
// needed : scc
struct two_sat {
    vector<vector<int> > adj;
    vector<bool> ans;

    two_sat(int n) : adj(2 * n), ans(n) {}

    void add_clause(int i, bool f, int j,
       bool g) {
        adj[2 * i + f].push_back(2 * j + !g);
        adj[2 * j + g].push_back(2 * i + !f);
    }

    bool vaild() {
        auto c = scc(adj);
        vector<int> id(adj.size());
        for (int i = 0; i < c.size(); i++)
            for (int j: c[i]) id[j] = i;
        for (int i = 0; i < ans.size(); i++)
           {
            if (id[i * 2] == id[i * 2 + 1])
                return false;
            ans[i] = id[i * 2] > id[i * 2 +
               1];
        }
        return true;
    }
};
```

3 Data Structure

3.1 BIT [d7b77a]

```
template<typename T>
struct bit {
    vector<T> t;

    bit(int n) : t(n + 1, 0) {}

    void modify(int idx, T val) {
        for (idx++; idx < t.size(); idx +=
           idx & (-idx))
            t[idx] += val;
    }

    T sum(int idx) {
        T r = 0;
        for (idx++; idx; idx -= idx & (-idx))
            r += t[idx];
        return r;
    }

    int lower_bound(T k) {
        int r = 0;
        for (int i = __lg(t.size()); i >= 0;
           i--) {
```

```
            int p = (1 << i) + r;
            if (p < t.size() && k - t[p] >=
                0)
                k -= t[p], r = p;
        }
        return r;
    }
};
```

3.2 DSU [57d5ce]

```
struct dsu {
    vector<int> p;

    dsu(int n) : p(n, -1) {}

    int par(int x) {
        return p[x] < 0 ? x : p[x] = par(p[x
            ]);
    }

    bool same(int a, int b) {
        return par(a) == par(b);
    }

    int size(int x) {
        return -p[par(x)];
    }

    bool unite(int x, int y) {
        x = par(x), y = par(y);
        if (x == y) return false;
        if (p[x] > p[y]) swap(x, y);
        p[x] += p[y], p[y] = x;
        return true;
    }
};
```

3.3 sparse table [09672e]

```
template<typename T, T (*f)(T, T)>
struct sparse_table {
    int lv, n;
    vector<vector<T> > t;

    sparse_table(vector<T> &a) : lv(__lg(a.
        size()) + 1), n(a.size()) {
        t = vector(lv, a);
        for (int i = 1; i < lv; i++)
            for (int j = 0; j + (1 << i - 1)
                < n; j++)
                t[i][j] = f(t[i - 1][j], t[i
                    - 1][j + (1 << i - 1)]);
    }

    T query(int l, int r) {
        int i = __lg(r - l + 1);
        return f(t[i][l], t[i][r - (1 << i) +
            1]);
    }
};
```

3.4 segment tree [91e61a]

```
template<typename T, T (*f)(T, T), T (*e)()>
struct segtree {
    int n;
    vector<T> t;

    segtree(int n) : n(n), t(2 * n, e()) {}

    segtree(vector<T> &a) : n(a.size()), t(2
        * n) {
        for (int i = 0; i < n; i++) t[i + n]
            = a[i];
        for (int i = n - 1; i > 0; i--)
            t[i] = f(t[i << 1], t[i << 1 |
                1]);
    }

    void modify(int p, T val) {
        for (t[p += n] = val; p >>= 1;)
            t[p] = f(t[p << 1], t[p << 1 |
                1]);
    }

    T query(int l, int r) {
        T ls = e(), rs = e();
        for (l += n, r += n + 1; l < r; l >>=
            1, r >>= 1) {
            if (l & 1) ls = f(ls, t[l++]);
            if (r & 1) rs = f(t[--r], rs);
        }
        return f(ls, rs);
    }
};
```

3.5 segment tree(lazy prop) [b5b477]

```
template<typename V, V (*f)(V, V), typename T
    , V (*mp)(T, V), T (*c)(T, T), T (*te)()>
struct segtree {
    struct node {
        V val;
        T tag;
        int l, r;
        node *lc, *rc;

        node(int l, int r) : l(l), r(r) {
            lc = rc = nullptr, tag = te();
        }

        void get(T new_tag) {
            val = mp(new_tag, val);
            tag = c(new_tag, tag);
        }

        void up() {
            if (lc && rc) val = f(lc->val, rc
                ->val);
        }

        void down() {
            if (lc && rc) lc->get(tag), rc->
                get(tag);
            tag = te();
        }
    } *root;
```

```
    node *build(int l, int r, const vector<V>
        &a) {
        node *p = new node(l, r);
        if (l == r) return p->val = a[l], p;
        int m = l + r >> 1;
        p->lc = build(l, m, a);
        p->rc = build(m + 1, r, a);
        return p->up(), p;
    }

    void modify(int l, int r, T t, node *p =
        nullptr) {
        if (!p) return modify(l, r, t, root);
        if (p->r < l || p->l > r) return;
        p->down();
        if (l <= p->l && p->r <= r) return p
            ->get(t);
        modify(l, r, t, p->lc), modify(l, r,
            t, p->rc);
        p->up();
    }

    V query(int l, int r, node *p = nullptr)
        {
        if (!p) return query(l, r, root);
        p->down();
        if (l <= p->l && p->r <= r) return p
            ->val;
        int m = p->l + p->r >> 1;
        if (r <= m) return query(l, r, p->lc)
            ;
        if (l > m) return query(l, r, p->rc);
        return f(query(l, r, p->lc), query(l,
            r, p->rc));
    }

    segtree(const vector<V> &a) {
        root = build(0, a.size() - 1, a);
    }
};
```

3.6 treap [d18c75]

```
// 1. remember deal with null pointer
// 2. call up() when constuct new node

struct node {
    int rev = 0, sz = 1, pri = rand();
    node *lc = nullptr, *rc = nullptr;

    void down() {
        if (!rev) return;
        swap(lc, rc);
        if (lc) lc->rev ^= 1;
        if (rc) rc->rev ^= 1;
        rev = 0;
    }

    void up() {
        sz = (lc ? lc->sz : 0) + (rc ? rc->sz
            : 0) + 1;
    }
};

node *merge(node *a, node *b) {
    if (!a || !b) return a ? a : b;
    if (a->pri < b->pri) {
        a->down();
        a->rc = merge(a->rc, b);
        a->up();
        return a;
    }
    b->down();
    b->lc = merge(a, b->lc);
    b->up();
    return b;
}

pair<node *, node *> split(node *t, int k) {
    if (!t) return {nullptr, nullptr};
    t->down();
    int sz = (t->lc ? t->lc->sz : 0);
    if (k <= sz) {
        auto [tl, tr] = split(t->lc, k);
        t->lc = tr;
        t->up();
        return {tl, t};
    }
    auto [tl, tr] = split(t->rc, k - sz - 1);
    t->rc = tl;
    t->up();
    return {t, tr};
}

array<node *, 3> split_range(node *t, int l,
    int r) {
    auto [lt, rp] = split(t, r + 1);
    auto [lp, mp] = split(lt, l);
    return {lp, mp, rp};
}
```

3.7 lichao segment tree [050df1]

```
template<typename T, bool get_min>
struct lichao {
    struct line {
        T a, b;

        line(T a, T b) : a(a), b(b) {}

        T operator()(T x) const { return a *
            x + b; }
    };

    int n;
    vector<line> fs;
    vector<T> xs;

    lichao(const vector<T> &x) : xs(x) {
        sort(xs.begin(), xs.end());
        xs.erase(unique(xs.begin(), xs.end())
            , xs.end());
        n = xs.size();
        fs.assign(n << 1, line(0,
            numeric_limits<T>::max()));
    }
```

```
    void update(T a, T b, int l, int r) {
        if (!get_min) a = -a, b = -b;
        line g(a, b);
        for (l += n, r += n; l < r; l >>= 1,
            r >>= 1) {
            if (l & 1) descend(g, l++);
            if (r & 1) descend(g, --r);
        }
    }

    void descend(line g, int i) {
        int l = i, r = i + 1;
        while (l < n) l <<= 1, r <<= 1;
        while (l < r) {
            int c = (l + r) >> 1;
            T xl = xs[l - n], xc = xs[c - n],
              xr = xs[r - 1 - n];
            line &f = fs[i];
            if (f(xl) <= g(xl) && f(xr) <= g(
                xr)) return;
            if (f(xl) >= g(xl) && f(xr) >= g(
                xr))
                return f = g, void();
            if (f(xc) > g(xc)) swap(f, g);
            if (f(xl) > g(xl)) i = i << 1 |
                0, r = c;
            else i = i << 1 | 1, l = c;
        }
    }

    void add_line(T a, T b) { update(a, b, 0,
        n); }

    void add_seg(T a, T b, T xl, T xr, bool
        closed = false) {
        int l = lower_bound(xs.begin(), xs.
            end(), xl) - xs.begin();
        int r = lower_bound(xs.begin(), xs.
            end(), xr) - xs.begin();
        if (closed) r = upper_bound(xs.begin
            (), xs.end(), xr) - xs.begin();
        update(a, b, l, r);
    }

    T get(T x) const {
        int i = lower_bound(xs.begin(), xs.
            end(), x) - xs.begin();
        assert(i != n && x == xs[i]);
        T y = numeric_limits<T>::max();
        for (i += n; i > 0; i >>= 1) y = min(
            y, fs[i](x));
        return get_min ? y : -y;
    }
};
```

3.8 binary trie [d3b22e]

```
template<typename T, int B>
struct btrie {
    vector<array<int, 2> > nxt{{0, 0}};
    vector<int> cnt{0};

    void insert(T num) {
        for (int i = B - 1, n = 0; i >= 0; i
            --) {
            int t = (num >> i) & 1;
            if (!nxt[n][t]) {
                nxt[n][t] = nxt.size();
                nxt.push_back({0, 0});
                cnt.push_back(0);
            }
            n = nxt[n][t], cnt[n]++;
        }
    }

    bool remove(T num) {
        if (!exist(num)) return false;
        for (int i = B - 1, n = 0; i >= 0; i
            --)
            n = nxt[n][(num >> i) & 1], cnt[n
                ]--;
        return true;
    }

    bool exist(T num, T x = 0) {
        for (int i = B - 1, n = 0; i >= 0; i
            --) {
            int t = ((num ^ x) >> i) & 1;
            if (!nxt[n][t] || !cnt[nxt[n][t
                ]])
                return false;
            n = nxt[n][t];
        }
        return true;
    }

    T find(T x = 0, bool mn = false) {
        for (int i = B - 1, n = 0; i >= 0; i
            --) {
            T t = (x >> i) & 1;
            if (mn) t ^= 1;
            if (nxt[n][1 ^ t] && cnt[nxt[n][1
                ^ t]])
                t ^= 1;
            n = nxt[n][t], x ^= t << i;
        }
        return x;
    }
};
```

4 Flow and Matching

4.1 bipartite matching [5607b5]

```
vector<pair<int, int> > hopcroft_karp(vector<
    pair<int, int> > e) {
    int n = -1, m = -1;
    shuffle(e.begin(), e.end(), mt19937(
        random_device{}()));
    for (auto &[u, v]: e)
        n = max(n, u + 1), m = max(m, v + 1);
    vector<int> deg(n + 1);
    for (auto &[u, v]: e) deg[u]++;
    partial_sum(deg.begin(), deg.end(), deg.
        begin());
    vector<int> g(e.size()), l(n, -1), r(m,
        -1), a, p, q(n);
```

```
    for (auto &[u, v]: e) g[--deg[u]] = v;
    while (true) {
        a.assign(n, -1), p.assign(n, -1);
        int t = 0;
        for (int i = 0; i < n; i++)
            if (l[i] == -1) q[t++] = a[i] = p
                [i] = i;
        bool match = false;
        for (int i = 0; i < t; i++) {
            int x = q[i];
            if (~l[a[x]]) continue;
            for (int j = deg[x]; j < deg[x +
                1]; j++) {
                int y = g[j];
                if (r[y] == -1) {
                    while (~y)
                        r[y] = x, swap(l[x],
                            y), x = p[x];
                    match = true;
                    break;
                }
                if (p[r[y]] == -1)
                    q[t++] = y = r[y], p[y] =
                        x, a[y] = a[x];
            }
        }
        if (!match) break;
    }
    vector<pair<int, int> > ret;
    for (int i = 0; i < n; i++)
        if (~l[i]) ret.emplace_back(i, l[i]);
    return ret;
}
```

4.2 hungarian algorithm [ff52a1]

```
template<typename T>
pair<T, vector<int> > hungarian(const vector<
    vector<int> > &a) {
    if (a.empty()) return {0, {}};
    int n = a.size() + 1, m = a[0].size() +
        1;
    vector<int> p(m), ans(n - 1);
    vector<T> u(n), v(m);
    for (int i = 1; i < n; i++) {
        p[0] = i;
        int now = 0;
        vector<int> pre(m, -1), vis(m + 1);
        vector<T> dis(m, numeric_limits<T>::
            max());
        do {
            vis[now] = true;
            int t = p[now], nxt;
            T d = numeric_limits<T>::max();
            for (int j = 1; j < m; j++)
                if (!vis[j]) {
                    T cur = -a[t - 1][j - 1]
                        - u[t] - v[j];
                    if (cur < dis[j]) dis[j]
                        = cur, pre[j] = now;
                    if (dis[j] < d) d = dis[j
                        ], nxt = j;
                }
            for (int j = 0; j < m; j++) {
                if (vis[j]) u[p[j]] += d, v[j
                    ] -= d;
                else dis[j] -= d;
            }
            now = nxt;
        } while (p[now]);
        while (now) {
            int t = pre[now];
            p[now] = p[t], now = t;
        }
    }
    for (int i = 1; i < m; i++)
        if (p[i]) ans[p[i] - 1] = i - 1;
    return {v[0], ans};
}
```

4.3 maximum flow [5b780c]

```
// time complexity : O(V^2E)
template<typename T>
struct dinic {
    struct edge {
        int u, v;
        T cap, flow = 0;

        edge(int u, int v, T c) : u(u), v(v),
            cap(c) {}
    };

    int n;
    vector<edge> e;
    vector<vector<int> > adj;
    T bound;

    dinic(int n) : n(n), adj(n), bound(0) {}

    void add_edge(int u, int v, T c) {
        adj[u].push_back(e.size());
        e.emplace_back(u, v, c);
        adj[v].push_back(e.size());
        e.emplace_back(v, u, 0);
        bound += c;
    }

    T flow(int s, int t) {
        vector<int> lv(n), p(n);
        auto bfs = [&]() {
            fill_n(lv.begin(), n, -1), lv[s]
                = 0;
            queue<int> q({s});
            while (!q.empty()) {
                for (int i: adj[q.front()]) {
                    auto &[u, v, c, f] = e[i
                        ];
                    if (f >= c || lv[v] !=
                        -1) continue;
                    lv[v] = lv[u] + 1, q.push
                        (v);
                }
                q.pop();
            }
            return lv[t] != -1;
        };
```

```
        function<T(int, T)> dfs = [&](int a,
            T up) {
            if (a == t) return up;
            T used = 0;
            for (int &i = p[a]; i < adj[a].
                size(); i++) {
                auto &[u, v, c, f] = e[adj[a
                    ][i]];
                if (f >= c || lv[u] + 1 != lv
                    [v]) continue;
                T d = dfs(v, min(up - used, c
                    - f));
                if (d == 0) continue;
                e[adj[u][i] ^ 0].flow += d;
                e[adj[u][i] ^ 1].flow -= d;
                used += d;
                if (used == up) break;
            }
            return used;
        };
        T ret = 0;
        while (bfs()) {
            fill_n(p.begin(), n, 0);
            while (T f = dfs(s, bound)) ret
                += f;
        }
        return ret;
    }
};
```

4.4 minimum cost flow [4be984]

```
template<typename T, typename C>
struct mcf {
    struct edge {
        int u, v;
        T cap, flow = 0;
        C cost;

        edge(int u, int v, T cap, C cost) : u
            (u), v(v), cap(cap), cost(cost)
            {}
    };

    int n;
    vector<edge> e;
    vector<vector<int> > adj;
    vector<C> dis, pot;
    vector<int> par;
    const C INF = numeric_limits<C>::max();

    mcf(int n) : n(n), adj(n), dis(n), pot(n)
        , par(n) {}

    void add_edge(int u, int v, T cap, C cost
        ) {
        adj[u].push_back(e.size());
        e.emplace_back(u, v, cap, cost);
        adj[v].push_back(e.size());
        e.emplace_back(v, u, 0, -cost);
    }

    bool dijkstra(int s, int t) {
        fill(dis.begin(), dis.end(), INF);
        dis[s] = 0;
        priority_queue<pair<C, int>, vector<
            pair<C, int> >, greater<> > q;
        q.push({0, s});
        while (!q.empty()) {
            auto [d, u] = q.top();
            q.pop();
            if (d > dis[u]) continue;
            for (int i: adj[u]) {
                auto &[u, v, cap, flow, cost]
                    = e[i];
                if (flow < cap && dis[v] >
                    dis[u] + cost + pot[u] -
                    pot[v]) {
                    dis[v] = dis[u] + cost +
                        pot[u] - pot[v];
                    par[v] = i;
                    q.push({dis[v], v});
                }
            }
        }
        return dis[t] != INF;
    }

    pair<T, C> flow(int s, int t, T lim =
        numeric_limits<T>::max()) {
        T flow = 0;
        C cost = 0;
        while (dijkstra(s, t) && lim - flow >
            0) {
            for (int i = 0; i < n; i++)
                if (dis[i] < INF) pot[i] +=
                    dis[i];
            T d = lim - flow;
            for (int u = t; u != s; u = e[par
                [u]].u)
                d = min(d, e[par[u]].cap - e[
                    par[u]].flow);
            for (int u = t; u != s; u = e[par
                [u]].u) {
                e[par[u] ^ 0].flow += d;
                e[par[u] ^ 1].flow -= d;
                cost += d * e[par[u]].cost;
            }
            flow += d;
        }
        return {flow, cost};
    }
};
```

4.5 bounded flow

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

$v \to T$ with an additional edge of capacity $-in(v)$.

5. The arbitrary flow only exists when all the edges from S are fully utilized.
   - To maximize/minimize flow from A to B, connect $B \to A$ with capacity ∞, and remove all additional edges. Then apply maximum flow on residual network from A to B or from B to A. The solution of each edge e is $l_e + f_e$

5 Geometry

5.1 2D point definition [b12177]

```
template<typename T>
struct point {
    T x, y;

    point(T x = 0, T y = 0) : x(x), y(y) {}

    template<typename O>
    operator point<O>() {
        return {x, y};
    }

    operator bool() {
        return abs(x) + abs(y) < 1e-9;
    }

    point operator+(const point &o) {
        return {x + o.x, y + o.y};
    }

    point operator-(const point &o) {
        return {x - o.x, y - o.y};
    }

    point operator*(const T &o) {
        return {x * o, y * o};
    }

    point operator/(const T &o) {
        return {x / o, y / o};
    }

    T operator*(const point &o) {
        return x * o.x + y * o.y;
    }

    T operator/(const point &o) {
        return x * o.y - y * o.x;
    }

    T norm() {
        return x * x + y * y;
    }

    long double arg() {
        return atan2l(y, x);
    }
};
```

5.2 convex hull [77794c]

```
template<typename T>
vector<point<T> > convex_hull(vector<point<T>
    > p) {
    sort(p.begin(), p.end(), [](auto &a, auto
        &b) {
        return a.x < b.x || (a.x == b.x && a.
            y < b.y);
    });
    p.erase(unique(p.begin(), p.end(), [](
        auto &a, auto &b) {
        return a - b;
    }), p.end());
    if (p.size() <= 1) return p;
    vector<point<T> > s(2 * p.size());
    int n = 0;
    for (int i = 0; i < 2; i++) {
        int l = n + 1;
        for (auto j: p) {
            while (n > l && (s[n - 1] - s[n -
                2]) / (j - s[n - 2]) <= 0)
                n--;
            s[n++] = j;
        }
        n--, reverse(p.begin(), p.end());
    }
    return s.resize(n), s;
}
```

5.3 polar sort [98f2ce]

```
template<typename T>
struct polar_sort {
    static bool up(point<T> &p) {
        return p.y > 0 || (!p.y && p.x < 0);
    }

    static bool cmp(point<T> &a, point<T> &b)
        {
        if (!is_integral_v<T>)
            return a.arg() < b.arg();
        if (up(a) == up(b))
            return !a && !b ? a / b > 0 : a <
                b;
        return up(a) < up(b);
    }
};
```

5.4 rotating calipers [c370f7]

```
// needed : convex hull
template<typename T>
auto rotating_calipers(vector<point<T> > &p)
    {
    auto h = convex_hull(p);
    auto nxt = [&](int v) { return (v + 1) %
        h.size(); };
    using info = pair<pair<point<T>, point<T>
        >, point<T> >;
```

```
    vector<info> ret(h.size());
    for (int i = 0, ptr = nxt(0); i < h.size
        (); i++) {
        while (abs((h[i] - h[nxt(i)]) / (h[
            nxt(i)] - h[ptr])) <
                abs((h[i] - h[nxt(i)]) / (h[
                    nxt(i)] - h[nxt(ptr)])))
            ptr = nxt(ptr);
        ret[i] = {{h[i], h[nxt(i)]}, h[ptr]};
    }
    return ret;
}
```

6 Graph

6.1 HLD [6c970e]

```
struct hld {
    vector<vector<int> > adj;
    vector<int> dep, top, pos, par, hvy, sz;

    hld(int n) : adj(n), par(n, -1), hvy(n,
        -1), sz(n, 1) {
        dep = top = pos = vector<int>(n);
    }

    void add_edge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    void build(int root = 0) {
        function<void(int)> init = [&](int u)
            {
            for (int &v: adj[u]) {
                if (v == par[u]) continue;
                par[v] = u, dep[v] = dep[u] +
                    1;
                init(v), sz[u] += sz[v];
                if (!~hvy[u] || sz[v] > sz[
                    hvy[u]])
                    hvy[u] = v;
            }
        };
        int t = 0;
        function<void(int, int)> sep = [&](
            int u, int h) {
            top[u] = h, pos[u] = t++;
            if (hvy[u] != -1) sep(hvy[u], h);
            for (int v: adj[u])
                if (v != par[u] && v != hvy[u
                    ])
                    sep(v, v);
        };
        init(root), sep(root, root);
    }

    int lca(int u, int v) {
        while (top[u] != top[v]) {
            if (dep[top[u]] < dep[top[v]])
                swap(u, v);
            u = par[top[u]];
        }
        return dep[u] > dep[v] ? v : u;
    }

    vector<pair<int, int> > find_path(int u,
        int v) {
        vector<pair<int, int> > l, r;
        while (top[u] != top[v]) {
            if (dep[top[u]] > dep[top[v]])
                l.emplace_back(u, top[u]), u
                    = par[top[u]];
            else
                r.emplace_back(top[v], v), v
                    = par[top[v]];
        }
        l.emplace_back(u, v);
        return l.insert(l.end(), r.rbegin(),
            r.rend()), l;
    }
};
```

6.2 SCC [cb5678]

```
vector<vector<int> > scc(vector<vector<int> >
    &adj) {
    int n = adj.size();
    vector<int> pos(n, -1), vis(n), stk;
    vector<vector<int> > res;
    function<int(int)> dfs = [&](int u) {
        int low = pos[u] = stk.size();
        stk.push_back(u);
        for (int v: adj[u])
            if (!vis[v])
                low = min(low, ~pos[v] ? pos[
                    v] : dfs(v));
        if (low == pos[u]) {
            res.emplace_back(stk.begin() +
                low, stk.end());
            for (int v: res.back()) vis[v] =
                true;
            stk.resize(low);
        }
        return low;
    };
    for (int i = 0; i < n; i++) if (!vis[i])
        dfs(i);
    return reverse(res.begin(), res.end()),
        res;
}
```

6.3 BECC [ec9ad5]

```
vector<vector<int> > becc(vector<vector<int>
    > &adj) {
    int n = adj.size(), t = 0;
    vector<int> dfn(n), low(n), stk;
    vector<vector<int> > res;
    function<void(int, int)> dfs = [&](int u,
        int p) {
        dfn[u] = low[u] = ++t;
        stk.push_back(u);
        for (int v: adj[u]) {
            if (v == p) { p = -1; continue; }
            if (dfn[v]) {
                low[u] = min(low[u], dfn[v]);
                continue;
```

```
            }
            dfs(v, u), low[u] = min(low[u],
                low[v]);
        }
        if (low[u] == dfn[u]) {
            res.push_back({});
            for (int w = -1; w != u; stk.
                pop_back())
                res.back().push_back(w = stk.
                    back());
        }
    };
    for (int i = 0; i < n; i++) if (!dfn[i])
        dfs(i, -1);
    return res;
}
```

6.4 BVCC [f6036c]

```
vector<vector<int> > bvcc(vector<vector<int>
    > &adj) {
    int n = adj.size(), t = 0;
    vector<int> dfn(n), low(n), stk;
    vector<vector<int> > res;
    function<void(int)> dfs = [&](int u) {
        dfn[u] = low[u] = t++;
        stk.push_back(u);
        if (adj[u].empty()) res.push_back({u
            });
        for (int v: adj[u]) {
            if (dfn[v]) {
                low[u] = min(low[u], dfn[v]);
                continue;
            }
            dfs(v), low[u] = min(low[u], low[
                v]);
            if (low[v] == dfn[u]) {
                res.push_back({u});
                for (int w = -1; w != v; stk.
                    pop_back())
                    res.back().push_back(w =
                        stk.back());
            }
        }
    };
    for (int i = 0; i < n; i++) if (!dfn[i])
        dfs(i);
    return res;
}
```

6.5 virtual tree [ea1911]

```
// needed : hld
vector<pair<int, int> > virtual_tree(hld &d,
    vector<int> p) {
    auto cmp = [&](int a, int b) {
        return d.pos[a] < d.pos[b];
    };
    sort(p.begin(), p.end(), cmp);
    for (int i = 1, n = p.size(); i < n; i++)
        p.push_back(d.lca(p[i - 1], p[i]));
    sort(p.begin(), p.end(), cmp);
    p.erase(unique(p.begin(), p.end()), p.end
        ());
    vector<pair<int, int> > vt;
    for (int i = 1; i < p.size(); i++)
        vt.emplace_back(d.lca(p[i - 1], p[i])
            , p[i]);
    return vt;
}
```

7 Math

7.1 linear sieve [28c486]

```
vector<int> sieve(int n) {
    vector<int> d(n), l;
    for (int i = 2; i < n; i++) {
        if (d[i] == 0)
            d[i] = i, l.push_back(i);
        for (int &j: l) {
            if (1ll * i * j >= n) break;
            d[i * j] = j;
            if (j == d[i]) break;
        }
    }
    return d;
}
```

7.2 type definition [d2ca96]

```
using i32 = int;
using i64 = long long;
using i128 = __int128;
using u32 = unsigned int;
using u64 = unsigned long long;
using u128 = unsigned __int128;
```

7.3 josephus [0c9a5b]

```
int josephus(int n, int k, int p) {
    int ans = (k - 1) % (n - p);
    for (int i = n - p + 1; i = n; i++)
        ans = (ans + k) % i;
    return ans;
}
```

7.4 floor sum [e2e580]

```
// return sum of floor((a * i + b) / m) for
    all i = 0 ~ n - 1
i64 floor_sum(i64 n, i64 m, i64 a, i64 b) {
    if (m == 0) return 0;
    i64 ans = a / m * n * (n - 1) / 2 + b / m
        * n;
    a %= m, b %= m;
    i64 y = (a * (n - 1) + b) / m;
    ans += y * (n - 1) - floor_sum(y, a, m, m
        - b - 1);
    return ans;
}
```

7.5 CRT [ce3bc8]

```
// needed : exgcd
pair<i64, i64> crt(const vector<pair<i64, i64
    > > &e) {
```

```
    i64 x = 0, l = 1;
    for (auto [r, m]: e) {
        r %= m;
        if (m > l) swap(x, r), swap(l, m);
        auto [a, b, g] = exgcd(l, m);
        if ((r - x) % g) return {0, 0};
        i64 n = m / g;
        x = (x + (r - x) / g * a % n * l) % (
            l * n);
        l *= n;
    }
    return {x < 0 ? x + l : x, l};
}
```

7.6 FFT [27ffa0]

```
void fft(vector<complex<double> > &a) {
    static vector<complex<long double> > r(2,
        {1, 0});
    static vector<complex<double> > rt(2, {1,
        0});
    int n = a.size(), l = 31 - __builtin_clz(
        n);
    for (static int k = 2; k < n; k *= 2) {
        r.resize(n), rt.resize(n);
        auto x = polar(1.0L, acos(-1.0L) / k)
            ;
        for (int i = k; i < 2 * k; i++)
            rt[i] = r[i] = i & 1 ? r[i / 2] *
                x : r[i / 2];
    }
    vector<int> rev(n);
    for (int i = 0; i < n; i++) {
        rev[i] = (rev[i / 2] | (i & 1) << l)
            / 2;
        if (i < rev[i]) swap(a[i], a[rev[i]])
            ;
    }
    for (int k = 1; k < n; k *= 2)
        for (int i = 0; i < n; i += 2 * k)
            for (int j = 0; j < k; j++) {
                auto z = rt[j + k] * a[i + j
                    + k];
                a[i + j + k] = a[i + j] - z,
                    a[i + j] += z;
            }
}
```

7.7 convolution(mod) [6fd829]

```
// needed : fft
template<int MOD = 998244353>
vector<i64> convolution(const vector<i64> &a,
    const vector<i64> &b) {
    if (a.empty() || b.empty()) return {};
    vector<i64> ret(a.size() + b.size() - 1);
    int n = 2 << __lg(ret.size()), cut = sqrt
        (MOD);
    vector<complex<double> > l(n), r(n), outs
        (n), outl(n);
    for (int i = 0; i < a.size(); i++)
        l[i] = {a[i] / cut, a[i] % cut};
    for (int i = 0; i < b.size(); i++)
        r[i] = {b[i] / cut, b[i] % cut};
    fft(l), fft(r);
    for (int i = 0; i < n; i++) {
        int j = -i & (n - 1);
        outl[j] = (l[i] + conj(l[j])) * r[i]
            / (2.0 * n);
        outs[j] = (l[i] - conj(l[j])) * r[i]
            / (2.0 * n) / 1i;
    }
    fft(outl), fft(outs);
    for (int i = 0; i < ret.size(); i++) {
        i64 av = llround(real(outl[i])), cv =
            llround(imag(outs[i]));
        i64 bv = llround(imag(outl[i])) +
            llround(real(outs[i]));
        ret[i] = ((av % MOD * cut + bv) % MOD
            * cut + cv) % MOD;
    }
    return ret;
}
```

7.8 convolution(without mod) [2ce869]

```
// needed : fft
vector<i64> convolution(const vector<i64> &a,
    const vector<i64> &b) {
    if (a.empty() || b.empty()) return {};
    vector<i64> ret(a.size() + b.size() - 1);
    int n = 2 << __lg(ret.size());
    vector<complex<double> > in(n), out(n);
    for (int i = 0; i < a.size(); i++) in[i].
        real(a[i]);
    for (int i = 0; i < b.size(); i++) in[i].
        imag(b[i]);
    fft(in);
    for (int i = 0; i < n; i++) in[i] *= in[i
        ];
    for (int i = 0; i < n; i++)
        out[i] = in[-i & (n - 1)] - conj(in[i
            ]);
    fft(out);
    for (int i = 0; i < ret.size(); i++)
        ret[i] = llround(imag(out[i]) / (4 *
            n));
    return ret;
}
```

7.9 exgcd [c61dc9]

```
array<i64, 3> exgcd(i64 a, i64 b) {
    if (b == 0) return {1, 0, a};
    auto [x, y, g] = exgcd(b, a % b);
    return {y, x - a / b * y, g};
}
```

7.10 primality test [f098c4]

```
static bool is_prime(u64 n) {
    if (n == 1 || n % 2 == 0) return n == 2;
    u64 d = (n - 1) >> __builtin_ctzll(n - 1)
        ;
    vector<u64> s{2, 325, 9375, 28178,
        450775, 9780504, 1795265022};
    if (n < 4759123141ll) s = {2, 7, 61};
    for (auto &a: s) {
```

```
        if (n <= a) return true;
        u64 p = a % n;
        if (p == 0) continue;
        u64 x = 1;
        for (u64 e = d; e; e >>= 1) {
            if (1 & e) x = u128(x) * p % n;
            p = u128(p) * p % n;
        }
        if (x == 1) continue;
        for (u64 t = d; x != n - 1; t <<= 1)
            {
            x = u128(x) * x % n;
            if (x == 1 || t == n - 1) return
                false;
        }
    }
    return true;
}
```

7.11 pollard rho [6e0329]

```
// needed : miller rabin
vector<u64> factorize(u64 n) {
    static mt19937_64 rng(random_device{}());
    vector<u64> ret;
    function<void(u64)> rho = [&](u64 n) {
        if (n == 1) return;
        if (is_prime(n)) return ret.push_back
            (n);
        u64 x = 0, y = 0, p = 1, g, c = 0;
        auto f = [&](u64 v) -> u64 {
            return (u128(v) * v + c) % n;
        };
        for (u64 i = 0; i & 127 || (g = gcd(p
            , n)) == 1; i++) {
            if (x == y)
                c = rng() % n, x = rng() % n,
                    y = f(x);
            u64 t = u128(p) * (x > y ? x - y
                : y - x) % n;
            if (t) p = t;
            x = f(x), y = f(f(y));
        }
        rho(n / g), rho(g);
    };
    return rho(n), sort(ret.begin(), ret.end
        ()), ret;
}
```

7.12 RREF [ad6c3f]

```
template<typename T>
int gauss(vector<vector<T> > &a, int m = -1)
    {
    if (a.empty()) return 0;
    int n = a.size(), r = 0;
    if (m == -1) m = a[0].size();
    for (int c = 0; c < m && r < n; c++) {
        int t = r;
        for (int i = r + 1; i < n; i++)
            if (abs(a[i][c]) > abs(a[t][c]))
                t = i;
        if (abs(a[t][c]) == 0) continue;
        swap(a[t], a[r]);
        T s = 1.0 / a[r][c];
        for (int i = 0; i < m; i++) a[r][i]
            *= s;
        for (int i = 0; i < n; i++)
            if (i != r) {
                T t = a[i][c];
                for (int j = 0; j < m; j++)
                    a[i][j] -= t * a[r][j];
            }
        r++;
    }
    return r;
}
```

7.13 RREF(mod 2) [1369e3]

```
template<typename T>
int gauss(vector<vector<T> > &a, int m = -1)
    {
    if (a.empty()) return 0;
    int n = a.size(), r = 0;
    if (m == -1) m = a[0].size();
    for (int c = 0; c < m && r < n; c++) {
        int t = r;
        for (int i = r + 1; i < n; i++)
            if (abs(a[i][c]) > abs(a[t][c]))
                t = i;
        if (abs(a[t][c]) == 0) continue;
        swap(a[t], a[r]);
        T s = 1.0 / a[r][c];
        for (int i = 0; i < m; i++) a[r][i]
            *= s;
        for (int i = 0; i < n; i++)
            if (i != r) {
                T t = a[i][c];
                for (int j = 0; j < m; j++)
                    a[i][j] -= t * a[r][j];
            }
        r++;
    }
    return r;
}
```

8 String

8.1 z algorithm [d3b0e4]

```
template<typename T>
vector<int> z_value(T &s) {
    vector<int> z(s.size());
    for (int i = 1, l = 0, r = 0; i < s.size
        (); i++) {
        if (i < r) z[i] = min(r - i, z[i - l
            ]);
        while (i + z[i] < s.size() && s[z[i]]
            == s[i + z[i]])
            z[i]++;
        if (i + z[i] > r) l = i, r = i + z[i
            ];
    }
    return z[0] = s.size(), z;
}
```

8.2 KMP [75c877]

```
template<typename T>
vector<int> kmp_nxt(T &s) {
    vector<int> nxt(s.size(), 0);
    for (int i = 1, j = 0; i < s.size(); nxt[
        i++] = j) {
        while (j > 0 && s[i] != s[j]) j = nxt
            [j - 1];
        if (s[i] == s[j]) j++;
    }
    return nxt;
}
```

8.3 manacher algorithm [153aea]

```
template<typename T>
vector<int> manacher(const T &s) {
    if (s.empty()) return {};
    string t(s.size() * 2 + 1, '.');
    for (int i = 0; i < s.size(); i++)
        t[i * 2 + 1] = s[i];
    int n = t.size(), m = 0, r = 0;
    vector<int> p(n);
    for (int i = 0; i < n; i++) {
        p[i] = r > i ? min(r - i, p[2 * m - i
            ]) : 1;
        while (0 <= i - p[i] && i + p[i] < n
            &&
               t[i - p[i]] == t[i + p[i]])
            p[i]++;
        if (i + p[i] > r) m = i, r = i + p[i
            ];
    }
    for (int i = 0; i < n; i++) p[i]--;
    return p;
}
```

8.4 rolling hash [ff05b3]

```
struct phash {
    const int m = 1e9 + 7, b1 = 999999937, b2
        = 999999929;
    vector<long long> h1{0}, h2{0}, p1{1}, p2
        {1};

    template<typename T>
    phash(const T &a) {
        for (int i = 0; i < a.size(); i++) {
            p1.emplace_back((p1.back() * b1)
                % m);
            p2.emplace_back((p2.back() * b2)
                % m);
            h1.emplace_back((h1.back() * b1 +
                a[i]) % m);
            h2.emplace_back((h2.back() * b2 +
                a[i]) % m);
        }
    }

    long long get_hash(int l, int r) {
        auto v1 = (h1[r + 1] - h1[l] * p1[r -
            l + 1]) % m;
        auto v2 = (h2[r + 1] - h2[l] * p2[r -
            l + 1]) % m;
        v1 = (v1 + m) % m, v2 = (v2 + m) % m;
        return (v1 << 31) + v2;
    }
};
```

8.5 SA [c814a2]

```
// time complexity : O(n lg^2 n)
template<typename T>
vector<int> suffix_array(T &s) {
    int n = s.size();
    vector<int> sa(n), rk = vector<int>(s.
        begin(), s.end()), tmp(n);
    iota(sa.begin(), sa.end(), 0);
    for (int k = 1; k < n; k *= 2) {
        auto cmp = [&](int x, int y) {
            if (rk[x] != rk[y]) return rk[x]
                < rk[y];
            int rx = x + k < n ? rk[x + k] :
                -1;
            int ry = y + k < n ? rk[y + k] :
                -1;
            return rx < ry;
        };
        sort(sa.begin(), sa.end(), cmp);
        tmp[sa[0]] = 0;
        for (int i = 1; i < n; i++)
            tmp[sa[i]] = tmp[sa[i - 1]] + (
                cmp(sa[i - 1], sa[i]) ? 1 :
                0);
        swap(tmp, rk);
    }
    return sa;
}
```

8.6 SAIS [6948d0]

```
template<typename T>
vector<int> suffix_array(T &str, int lim =
    128) {
    vector<int> s(str.size() + 1);
    for (int i = 0; i < str.size(); i++)
        s[i] = str[i];
    function<vector<int>(vector<int> &, int)>
        sais = [&](auto &s, int k) {
        int n = s.size();
        vector<int> a(n), t(n), lms(n), cnt(k
            );
        cnt[s[n - 1]] = t[n - 1] = 1;
        for (int i = n - 2; i >= 0; cnt[s[i
            --]]++)
            t[i] = s[i] == s[i + 1] ? t[i +
                1] : s[i] < s[i + 1];
        partial_sum(cnt.begin(), cnt.end(),
            cnt.begin());
        int pos = 0;
        for (int i = 1; i < n; i++)
            if (t[i - 1] == 0 && t[i] == 1)
                t[i] = 2, lms[pos++] = i;
        auto induce = [&](int offset) {
            vector<int> p(k + 1);
```

```
            copy(cnt.begin(), cnt.end(), p.
                begin());
            auto push_left = [&](int n) { a
                [--p[s[n]]] = n; };
            auto push_right = [&](int n) { a[
                p[s[n]]++] = n; };
            for (int i = pos - 1; i >= 0; i
                --)
                push_left(lms[i + offset]);
            copy(cnt.begin(), cnt.end(), p.
                begin() + 1);
            for (int i = 0; i < n; i++)
                if (a[i] - 1 >= 0 && t[a[i] -
                    1] == 0)
                    push_right(a[i] - 1);
            copy(cnt.begin(), cnt.end(), p.
                begin());
            for (int i = n - 1; i >= 0; i--)
                if (a[i] - 1 >= 0 && t[a[i] -
                    1] != 0)
                    push_left(a[i] - 1);
        };
        induce(0);
        auto cmp = [&](int i, int j) {
            if (s[i] == s[j])
                while (s[++i] == s[++j] && t[
                    i] == t[j])
                    if (t[i] == 2 && t[j] ==
                        2)
                        return true;
            return false;
        };
        int new_k = 0;
        vector<int> new_s(n, -1);
        new_s.back() = 0;
        for (int i = 1, last = n - 1; i < n;
            i++) {
            if (t[a[i]] != 2) continue;
            if (!cmp(last, a[i])) new_k++;
            new_s[last = a[i]] = new_k;
        }
        new_s.erase(remove(new_s.begin(),
            new_s.end(), -1), new_s.end());
        vector<int> new_a;
        if (++new_k == pos) {
            new_a.resize(pos, -1);
            for (int i = 0; i < pos; i++)
                new_a[new_s[i]] = i;
        } else
            new_a = sais(new_s, new_k);
        for (int i = 0; i < pos; i++)
            lms[i + pos] = lms[new_a[i]];
        memset(&a[0], 0, n << 2);
        induce(pos);
        return a;
    };
    auto ans = sais(s, lim);
    return ans.erase(ans.begin()), ans;
}
```

8.7 LCP [5d7802]

```
template<typename T>
vector<int> lcp(T &s, const vector<int> &sa)
    {
    int n = s.size(), k = 0;
    vector<int> lcp(n), rk(n);
    for (int i = 0; i < n; i++) rk[sa[i]] = i
        ;
    for (int i = 0; i < n; i++, k ? k-- : 0)
        {
        if (rk[i] == n - 1) {
            k = 0;
            continue;
        }
        int j = sa[rk[i] + 1];
        while (i + k < n && j + k < n && s[i
            + k] == s[j + k])
            k++;
        lcp[rk[i]] = k;
    }
    return lcp[n - 1] = 0, lcp;
}
```

9 Theorem

9.1 fibonacci

$$F_{n-1}F_{n+1} - F_n^2 = (-1)^n$$
$$F_{n+k} = F_kF_{n+1} + F_{k-1}F_n$$
$$gcd(F_m, F_n) = F_{gcd(m,n)}$$

9.2 catalan number

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

9.3 pick theorem

$$A = I + \frac{B}{2} - 1$$

9.4 lucas theorem

$$\binom{N}{K} \equiv \prod_{i=0}^{x}\binom{n_i}{k_i} (mod\ P)$$

10 Python

```
# input
n = int(input())
l = [int(x) for x in input().split()]

# EOF
while True:
    try:
        input()
    except:
        break

# output
print([1, 2, 3], [4, 5, 6], sep=' ', end='')
print(' '.join(str(x) for x in [1, 2, 3]))
print(f'{0.111111: .3f}')

# ascii
```

```
ord('c')
chr(97)

# list
l = [x for x in range(3)]
l.append(4)

# sort
l.sort(reverse=True)
sorted(l, key=lambda x: x)

# dict
d = {1: "a"}
d['2'] = '3'
for k, v in d.items():
    print(k, v)
1 in d

# deque
from collections import deque
q = deque([3, 4, 5])
q.appendleft(2)
q.append(6)
q.popleft()
q.pop()
q[0]

# random
from random import *
randint(1, 10)
choice([1, 2, 3])
shuffle([1, 2, 3])

# Decimal
from fractions import Fraction
from decimal import Decimal, getcontext
getcontext().prec = 250
d = Decimal('0.3') # check -0 when output
f = Fraction(d)
f.numerator
f.denominator

# fast input
import io, os, sys
input = sys.stdin.readline
# call input().decode() to convert into
   string
input = io.BytesIO(os.read(0, os.fstat(0).
   st_size)).readline

# fast output
import sys
sys.stdout.write("need to be string")
```