Структуры данных

1. Декартово дерево

```
struct Node {
   int x;
   int y;
   int sz;
   bool rev;
   Node* l;
   Node* r;
   Node() {};
};

typedef Node* treap;
typedef pair<treap, treap> ptt;

treap newNode(int x, int y, int sz, Node* l, Node*
r) {
   treap t = new Node();
   t->x = x;
   t->y = y;
   t->sz = sz;
   t->rev = 0;
   t->l = l;
   t->r = r;
   return t;
}
treap createNode(int x) {
   return newNode(x, rnd(), 1, NULL, NULL);
}
int getSize(treap t) {
   if (! t) return 0;
   return t->sz;
}
treap fix (treap t) {
   if (t) t->sz = 1 + getSize(t->l) + getSize(t->r);
   return t;
}
treap rev(treap t) {
   if (t) t->rev ^= 1;
   return t;
}
treap push(treap t) {
   if (! t) return t;
   if (t->rev) {
      t->l = rev(t->l);
      t->r = rev(t->r);
      swap(t->l, t->r);
      t->rev = 0;
   }
   return fix(t);
}
treap merge(treap a, treap b) {
   if (! a) return b;
   if (! b) return a;
   a = push(a);
   b = push(b);
   if (a->y > b->y) {
      a->r = merge(a->r, b);
      return fix(a);
   }
   else {
      b->l = merge(a, b->l);
      return fix(b);
   }
}
ptt splitK(treap t, int k) {
   if (getSize(t) <= k) return {t, NULL};
   if (k == 0) return {NULL, t};
   t = push(t);
   int szl = getSize(t->l);
   if (szl < k) {
      ptt p = splitK(t->r, k - szl - 1);
      t->r = p.first;
      return {fix(t), p.second};
   }
   else {
      ptt p = splitK(t->l, k);
      t->l = p.second;
      return {p.first, fix(t)};
   }
}
ptt split(treap t, int x) {
   if (! t) return {t, t};
   if (t->x < x) {
      ptt p = split(t->r, x);
      t->r = p.first;
      return {fix(t), p.second};
   }
   else {
      ptt p = split(t->l, x);
      t->l = p.second;
      return {p.first, fix(t)};
   }
}
treap insert(treap t, int x) {
   ptt p = split(t, x);
   treap q = createNode(x);
   return merge(p.first, merge(fix(q), p.second));
```

```
}
li sumltX(treap t, int x) {
   if (! t) return 0ll;
   if (t->x < x) return t->x + getSum(t->l) +
sumltX(t->r, x);
   else return sumltX(t->l, x);
}
bool contains(treap t, int x) {
   if (! t) return false;
   if (t->x == x) return true;
   if (t->x > x) return contains(t->l, x);
   else return contains(t->r, x);
}
treap revSeg(treap t, int l, int r) {
   ptt p = splitK(t, l);
   ptt q = splitK(p.second, r - l);
   q.first = rev(q.first);
   t = merge(p.first, merge(q.first, q.second));
   return fix(t);
}
treap createfromVector(const vector<int> &a) {
   treap t = NULL;
   for (auto x: a) {
      t = merge(t, createNode(x));
   }
   return t;
}
void getVector(treap t, vector<int> &a) {
   if (! t) return;
   t = push(t);
   getVector(t->l, a);
   a.push_back(t->x);
   getVector(t->r, a);
}
```

2. Дерево отрезков

```
void push(int v) {
   if (op[v]) {
      op[2 * v + 1] += op[v];
      op[2 * v + 2] += op[v];
      tr[v] += op[v];
      op[v] = 0;
   }
}
void recalc(int v, int l, int r) {
   push(v);
   int m = (l + r) / 2;
   tr[v] = tr[2 * v + 1] + tr[2 * v + 2];
}
void build(int v, int l, int r) {
   op[v] = 0ll;
   if (l == r - 1) {
      tr[v] = a[l];
      return;
   }
   int m = (l + r) / 2;
   build(v * 2 + 1, l, m);
   build(v * 2 + 2, m, r);
   recalc(v, l, r);
}

void upd_seg(int v, int l, int r, int lf, int rg, int val)
{
   if (l == lf && r == rg) {
      op[v] += val;
      return;
   }
   else if (lf >= rg) return;
   else {
      push(v);
      int m = (l + r) / 2;
      upd_seg(2 * v + 1, l, m, lf, min(rg, m), val);
      upd_seg(2 * v + 2, m, r, max(lf, m), rg, val);
      recalc(v, l, r);
   }
}

li query(int v, int l, int r, int lf, int rg) {
   push(v);
   if (lf == l && rg == r) return tr[v];
   else if (lf >= rg) return 0ll;
   int m = (l + r) / 2;
   return query(v * 2 + 1, l, m, lf, min(rg, m)) +
query(v * 2 + 2, m, r, max(lf, m), rg);
}
```

2.1. Дерево отрезков #2

```
 void build(int v, int l, int r) {
   if (l == r - 1) {
      T[v] = a[l];
      return;
   }
   int m = (l + r) / 2;
   build(2 * v + 1, l, m);
   build(2 * v + 2, m, r);
   T[v] = T[2 * v + 1] + T[2 * v + 2];
}
```

```
void push(int v, int l, int r) {
    if (l != r - 1) {
        ps[2 * v + 1] += ps[v];
        ps[2 * v + 2] += ps[v];
    }
    T[v] += ps[v] * (r - l);
    ps[v] = 0;
}
void upd(int v, int l, int r, int L, int R, int val) {
    push(v, l, r);
    if (L >= R) return;
    if (l == L && r == R) {
        ps[v] = val;
        push(v, l, r);
        return;
    }
    int m = (l + r) / 2;
    upd(2 * v + 1, l, m, L, min(m, R), val);
    upd(2 * v + 2, m, r, max(m, L), R, val);
    T[v] = T[2 * v + 1] + T[2 * v + 2];
}
int query(int v, int l, int r, int pos) {
    push(v, l, r);
    if (l == r - 1) return T[v];
    int m = (l + r) / 2;
    int res = 0;
    if (pos < m) {
        res = query(2 * v + 1, l, m, pos);
        push(2 * v + 2, m, r);
    }
    else {
        res = query(2 * v + 2, m, r, pos);
        push(2 * v + 1, l, m);
    }
    T[v] = T[2 * v + 1] + T[2 * v + 2];
    return res;
}
// Спуск по ДО
int trav(int v, int l, int r, int x) {
    if (l == r - 1) return T[v] >= x ? l : -1;
    int m = (l + r) / 2;
    if (T[2 * v + 1] >= x) return trav(2 * v + 1, l,
m, x);
    else return trav(2 * v + 2, m, r, x);
}
```

Подсчёт количества инверсий (вставить в MergeSort)

```
int inverse = 0;
vector<int> ans;
int i = 0, j = 0;
while (i < left.size() && j < right.size()) {
    if (left[i] <= right[j]) {
        ans.push_back(left[i]);
        i++;
    }
    else {
        ans.push_back(right[j]);
        inverse += (left.size() - i);
        j++;
    }
}
while (i < left.size()) {
    ans.push_back(left[i]);
    i++;
}
while (j < right.size()) {
    ans.push_back(right[j]);
    j++;
}
```

3. Очередь с приоритетами

```
priority_queue<pair<li, int>, vector<pair<li,
int>>, greater<pair<li, int>> q; // C++

import heapq
heapq.heappush(arr, val)
ans = heapq.heappop(arr) // Python
```

4. DSU

```
class DSU:
    def __init__(self, n):
        self.p = [i for i in range(n)]
        self.size = [1] * n

    def get_leader(self, x):
        if x != self.p[x]:
            self.p[x] = self.get_leader(self.p[x])
        return self.p[x]

    def merge(self, x, y):
        x = self.get_leader(x)
        y = self.get_leader(y)
        if x == y: return
        if self.size[x] > self.size[y]:
            x, y = y, x
        self.p[x] = y
```

```
        self.size[y] += self.size[x]
```

Строковые алгоритмы

1. Бор

```
int nxt[N][A];
bool term[N];
int sz = 1;
int createNode() {
    for (int i = 0; i < A; ++i) {
        nxt[sz][i] = -1;
    }
    term[sz] = false;
    return sz++;
}
void init() {
    sz = 0;
    createNode();
}
void addString(string s) {
    int cur = 0;
    for (auto& c : s) {
    if (nxt[cur][c - '0'] == -1)
        nxt[cur][c - '0'] = createNode();
        cur = nxt[cur][c - '0'];
    }
    term[cur] = false;
}
li fXor(li x) {
    li y = x;
    int cur = 0;
    for (int i = L - 1; ~i; --i) {
        li bit = (x >> i) & 1;
        if (nxt[cur][bit ^ 1] != -1) bit ^= 1;
        y ^= (bit << i);
        cur = nxt[cur][bit];
    }
    return y;
}
```

2. Ахо-Корасик

```
int go(int v, int c) {
    if (nodes[v].go[c] != -1) return nodes[v].go[c];
    if (nodes[v].nxt[c] != -1)
        return nodes[v].go[c] = nodes[v].nxt[c];
    if (v == 0)
        return nodes[v].go[c] = 0;
    return nodes[v].go[c] = go(suf(v), c);
}
int suf(int v) {
    if (nodes[v].suf != -1)
        return nodes[v].suf;
    if (v == 0 || nodes[v].p == 0)
        return nodes[v].suf = 0;
    return nodes[v].suf = go(suf(nodes[v].p),
nodes[v].pch);
}
int ssuf(int v) {
    if (nodes[v].ssuf != -1)
        return nodes[v].ssuf;
    if (v == 0 || nodes[v].p == 0)
        return nodes[v].ssuf = 0;
    int s = suf(v);
    if (nodes[s].term)
        return nodes[v].ssuf = s;
    return nodes[v].ssuf = ssuf(s);
}
```

3. Префикс-функция

```
vector<int> prefix_function(string s) {
    int n = (int)s.length();
    vector<int> pi(n);
    for (int i = 1; i < n; i++) {
        int j = pi[i - 1];
        while (j > 0 && s[i] != s[j])
            j = pi[j - 1];
        if (s[i] == s[j]:
            j++;
        pi[i] = j;
    }
    return pi;
}
// n – pr[n - 1] - период строки
```

4. Z-функция

```
s = input().strip()
n = len(s)
z = [0] * n
l, r = 0, 0
for i in range(1, n):
    j = max(0, min(z[i - l], r - i))
    while i + j < n and s[j] == s[i + j]:
        j += 1
    z[i] = j
    if i + j > r:
```

```
        l = i
        r = i + j
```

5. КМП-автомат

```
def kmp(s):
    n = len(s)
    pr = prefix_function(s)
    aut = [[0] * 26 for _ in range(n + 1)]
    for i in range(n + 1):
        for j in range(26):
            if i < n and s[i] == chr(ord('A') + j):
                aut[i][j] = i + 1
            elif i == 0:
                aut[i][j] = 0
            else:
                aut[i][j] = aut[pr[i - 1]][j]
    return aut
```

6. Хэши

```
const int P = 31;
const int M = 998244353;

vector<li> hash(string s) {
    int n = s.size();
    vector<li> f(n + 1, 0);
    for (int i = n - 1; i >= 0; i--) {
        f[i] = add(M, add(mul(f[i + 1],P),s[i]-'a'+1));
    }
    return f;
}
// Хэш подстроки с j по j + k символ
li sub = add(add(M, f[j]), -mul(f[j + k],
mod_pow(P, k)));
```

XOR-хэши

```
vector<int> b = a;
sort(b.begin(), b.end());
b.erase(unique(b.begin(), b.end()), b.end());
for (int i = 0; i < n; i++) a[i] =
lower_bound(b.begin(), b.end(), a[i]) - b.begin();
set<int> used;
used.insert(0);
for (int i = 0; i < int(b.size()); i++) {
    int x = rnd();
    while (used.count(x)) x = rnd();
    b[i] = x;
    used.insert(x);
}
for (int i = 0; i < n; i++) a[i] = b[a[i]];
vector<int> pr(n + 1, 0);
for (int i = 0; i < n; i++) pr[i + 1] = pr[i] ^ a[i];
```

Динамика

1. НВП

```
for (int i = 0; i < n; ++i) {
    auto it = lower_bound(ls.begin(), ls.end(),v[i]);
    if (it == ls.end())
        ls.push_back(v[i]);
   else
        *it = v[i];
  }
  cout << ls.size() << '\n';
```

2. Оптимизация рюкзака битсетами

```
bitset<N> dp;
dp[0] = 1;
for (int i = 0; i < n; ++i) {
    auto dp2 = dp;
    dp2 |= dp << a[i];
    dp ^= dp2;
    for (int s = dp._Find_first(); s < dp.size(); s =
dp._Find_next(s))
        p[s] = i;
    dp = dp2;
}
int pr = p[s];
vector<int> path;
while (~pr) {
    path.push_back(pr + 1);
    s -= a[p[s]];
    pr = p[s];
}
```

Графы

1. BFS

```
int d[N];
queue<int> q;
q.push(0);
while (! q.empty()) {
    int v = q.front();
```

```
    q.pop();
    for (int u: g[v]) {
        if (d[u] == INF) {
            d[u] = d[v] + 1;
            q.push(u);
        }
    }
}
```

2. Дейкстра

```
li d[N];
bool used[N];
priority_queue<pair<li, int>, vector<pair<li,
int>>, greater<pair<li, int>> q;
d[0] = 0;
q.push({0, 0});
while (! q.empty()) {
     int v = q.top().second;
    q.pop();
    if (used[v]) continue;
    used[v] = true;
    for (auto e: g[v]) {
        if (d[e.first] > d[v] + e.second) {
            d[e.first] = d[v] + e.second;
            q.push({d[e.first], e.first});
        }
    }
}
```

3. Конденсация графа

```
void ts(int v) {
    used[v] = true;
    for (auto u: g[v]) {
        if (! used[u]) ts(u);
    }
    ord.push_back(v)
}
void dfs(int v, int u) {
    comp[v] = u;
    for (auto u: tg[v]) {
        if (comp[u] == -1) dfs(u, k);
    }
}
for(int ii = 0; ii < n; ii++) {
    if (! used[i]) ts(i);
}
reverse(ord.begin(), ord.end());
int cnt = 1;
for (auto v: ord) {
     if (comp[v] == -1) dfs(v, cnt++);
}
```

4. Цикл в орграфе

```
bool dfs(int v, vector<vector<int>> &g,
vector<int> &p, vector<int> &used, int &c_st, int
&c_end) {
    used[v] = 1;
    for (int u : g[v]) {
        if (! used[u]) {
            p[u] = v;
            if (dfs(u,g,p,used,c_st,c_end)) return true;
        }
        else if (used[u] == 1) {
            c_st = u;
            c_end = v;
            return true;
        }
    }
    used[v] = 2;
    return false;
}
void solve() {
    int c_st = -1, c_end = -1;
    for (int v = 0; v < n; v++) {
        if (! used[v]) {
            dfs(v, g, p, used, c_st, c_end);
            if (c_st != -1 && c_end != -1) {
                vector<int> cycle;
                while (c_st != c_end) {
                    cycle.push_back(c_end);
                    c_end = p[c_end];
                }
                cycle.push_back(c_end);
                reverse(cycle.begin(), cycle.end());
            }
        }
    }
}
```

5. Мосты

```
int dfs(int v) {
    used[v] = true;
    int ans = INF;
    for (auto e: g[u]) {
        int to = v ^ e.v ^ e.u;
        if (! used[to]) {
```

```
            d[to] = d[v] + 1;
            int mn = dfs(to);
            if (mn > d[v]) is_bridge[e.i] = true;
            ans = min(mn, ans);
        }
        else if (u != p) ans = min(ans, d[u]);
    return ans;
}
```

6. Форд-Беллман

```
d[0] = 0;
for (int i = 0; i < N – 1; i++) {
    for (auto [v, u, w]: e) {
        d[u] = min(d[u], d[v] + w);
    }
}
```

// Поиск циклов отрицательного веса: выполним ещё одну итерацию, если на N-ной итерации что-то поменялось, существует цикл отрицательного веса.

7. Флойд

```
for (int i = 0; i < n; i++) {
    for (int v = 0; v < n; v++) {
        for (int u = 0; u < n; u++) {
            d[u][v] = min(d[u][v], d[u][i] + d[i][v]);
        }
    }
}
```

8. Потоки

```
class Edge:
    def __init__(self, to, w, f):
        self.to = to
        self.w = w
        self.f = f

def add_edge(fr, to, w):
    g[fr].append(len(e))
    e.append(Edge(to, w, 0))
    g[to].append(len(e))
    e.append(Edge(fr, 0, 0))

def rem(edge):
    return edge.w - edge.f

def dfs(v, f, k):
    if used[v]:
        return 0
    used[v] = True
    if v == T:
        return f
    for idx in g[v]:
        a = e[idx]
        r = rem(a)
        if r < k:
            continue
        pushed = dfs(a.to, min(r, f), k)
        if pushed:
            a.f += pushed
            e[idx ^ 1].f -= pushed
            return pushed
    return 0

S = 0
T = n - 1
ans = 0
for k in range(30, -1, -1):
    min_flow = 2 ** k
    while True:
        used = [False] * n
        flow = dfs(S, INF, min_flow)
        if not flow:
            break
        ans += flow
print(ans)
```

9. Min cost – max flow

```
struct edge {
    int y, cap, flow, cost;
    edge(int y=0, int cap=0, int flow=0,int cost=0)
: y(y), cap(cap), flow(flow), cost(cost) {};
};
vector<edge> e;
vector<int> g[N];

int residual(int x) return e[x].cap — e[x].flow;

void add_edge(int x, int y, int cap, int cost) {
    g[x].push_back(e.size());
    e.push_back(edge(y, cap, 0, cost));
    g[y].push_back(e.size());
    e.push_back(edge(x, 0, 0, -cost));
}
```

```
int s, int t, int V;
long long d[N];
int p[N];
int pe[N];

pair<int, long long> augment() {
queue<int> q;
q.push(s);
for (int i = 0; i < V; i++) {
   d[i] = INF;
   p[i] = -1;
   pe[i] = -1;
}
d[s] = 0;
vector<bool> in_queue(V);
in_queue[s] = true;
while (!q.empty()) {
    int k = q.front();
    q.pop();
    in_queue[k] = false;
    for (auto i : g[k]) {
       if (residual(i) == 0) continue;
       int y = e[i].y;
       int w = e[i].cost;
       if (d[y] > d[k] + w) {
          d[y] = d[k] + w;
          p[y] = k;
          pe[y] = i;
          if (!in_queue[y]) {
             q.push(y);
             in_queue[y] = true;
          }
       }
}
}
if (p[t] == -1) return { 0, 0ll };
int flow = 1e9;
int cur = t;
while (cur != s) {
   flow = min(flow, residual(pe[cur]));
   cur = p[cur];
}
cur = t;
while (cur != s) {
    e[pe[cur]].flow += flow;
    e[pe[cur] ^ 1].flow -= flow;
    cur = p[cur];
}
return { flow, d[t] };
}
```

```
pair<int, long long> mincost_maxflow() {
   int total_flow = 0;
   long long total_cost = 0ll;
   while (true) {
      pair<int, long long> f = augment();
      if (f.first != 0) {
         total_flow += f.first;
         total_cost += f.second * f.first;
      }
      else break;
}
return { total_flow, total_cost };
}
```

10. Паросочетания (минимальное покрытие путями)

```
int mt[N];
bool used[N];
int n, m;

bool dfs(int v, vector<vector<int>>& g) {
   if (used[v]) return false;
   used[v] = true;
   for (int u : g[v]) {
      if (mt[u] == -1 || dfs(mt[u], g)) {
         mt[u] = v;
         return true;
      }
   }
   return false;
}
void solve() {
   for (int i = 0; i < N; i++) mt[i] = -1;
   for (int i = 0; i < n; i++) {
      for (int i = 0; i < N; i++) used[i] = false;
      dfs(i, g);
   }
   int kl = 0;
   for (int i = 0; i < N; i++) {
      if (mt[i] != -1) {
         kl++;
      }
   }
   cout << n - kl << '\n';
   vector<vector<int>> paths(n, vector<int> (0));
   for (int i = 0; i < n; i++) {
      int cur = i;
      vector<int> ans;
```

```
        while (cur != -1) {
            ans.push_back(cur + 1);
            cur = mt[cur];
        }
        reverse(ans.begin(), ans.end());
        if (paths[ans[0] - 1].size() < ans.size()) {
            paths[ans[0] - 1] = ans;
        }
    }
}
```

Sqrt-декомпозиция

1. RSQ

```
sq = int(n ** 0.5)
b = [0] * (sq + 2)
for i in range(n):
    b[i // sq] += a[i]
for _ in range(m):
    q = [int(x) for x in input().split()]
    if q[0] == 0:
        t, l, r = q
        l -= 1
        r -= 1
        sm = 0
        while l <= r:
            if l % sq == 0 and l + sq <= r:
                sm += b[l // sq]
                l += sq
            else:
                sm += a[l]
                l += 1
        print(sm)
    else:
        t, i, x = q
        i -= 1
        b[i // sq] += (x - a[i])
        a[i] = x
```

2. Алгоритм Мо

```
void solve() {
    int sq = sqrt(n);
    auto comp = [&](pair<pair<int, int>, int> a,
pair<pair<int, int>, int> b) {
        pair<int, int> a1 = a.first, b1 = b.first;
        int x1 = a1.first / sq, x2 = b1.first / sq, y1
a1.second, y2 = b1.second;
        return x1 < x2 || x1 == x2 && y1 < y2;
    };
    sort(qs.begin(), qs.end(), comp);
    vector<int> cnt(b.size(), 0);
    int l = 0;
    int r = -1;
    int d = 0;
    vector<int> ans(q);
    for (int i = 0; i < q; i++) {
        int L = qs[i].first.first, R =
qs[i].first.second, j = qs[i].second;
        while (l > L) {
            if (++cnt[a[--l]] % 2) d++;
            else d--;
        }
        while (r < R) {
            if (++cnt[a[++r]] % 2) d++;
            else d--;
        }
        while (l < L) {
            if (--cnt[a[l++]] % 2) d++;
            else d--;
        }
        while (r > R) {
            if (--cnt[a[r--]] % 2) d++;
            else d--;
        }
}
```

Сжатие координат

```
vector<int> b = a;
sort(b.begin(), b.end());
b.erase(unique(b.begin(), b.end()), b.end());
for (int i = 0; i < n; i++)
    a[i] = lower_bound(b.begin(), b.end(), a[i]) —
b.begin();
}
```

Математика и ТЧ

1. FFT

```
from math import pi, e, log

def FFT(P, inverse=0):
    n = len(P)
    if n == 1:
        return P
    p_e = FFT(P[::2], inverse)
    p_o = FFT(P[1::2], inverse)
```

```
    w = e ** ((-1) ** (inverse + 1) * 2j * pi / n)
    y = [0] * n
    mod = 1
    for j in range(n // 2):
        y[j] = (p_e[j] + p_o[j] * mod) / (inverse + 1)
        y[j + n // 2] = (p_e[j] - p_o[j] * mod) /
(inverse + 1)
        mod *= w
    return y

x = FFT(a)
y = FFT(b)
for i in range(MAXN):
    y[i] *= x[i]
res = FFT(y, inverse=1)
for i in range(MAXN):
    cnt = round(res[i].real)
```

2. Модульная арифметика

```
int add(int n, int m) {
    n += m;
    while (n >= mod) n -= mod;
    while (n < 0) n += mod;
    return n;
}

int mul(int n, int m) {
    return (n * 1ll * m) % mod;
}

int mod_pow(int base, int n) {
     li ans = 1;
     while (n != 0) {
         if (n & 1) ans = mul(ans, base);
         n >>= 1;
         base = mul(base, base);
    }
    return ans;
}

int inv(int x) {
    return mod_pow(x, mod – 2);
}
```

3. Диофантовы уравнения

```
int exgcd(int a, int b, li& x1, li& y1) {
    if (b == 0) {
        x1 = 1;
        y1 = 0;
        return a;
    }
    li x, y;
    int d = exgcd(b, a % b, x, y);
    x1 = y;
    y1 = x - (a / b) * y;
    return d;
}

int d = exgcd(a, b, x, y);
if (c % d != 0) cout << "NO" << '\n';
else {
    cout << "YES" << '\n';
    int m = c / d;
    cout << x * m << y * m;
}

// Обратный элемент по любому модулю
li inverse(int a, int mod) {
    li x = 0, y = 0;
    int d = exgcd(a, mod, x, y);
    return x;
}
```

4. КТО

```
def CRT(a, m):
    n = len(a)
    M = 1
    for i in range(n):
        M *= m[i]
    z = [M // m[i] for i in range(n)]
    y = [(z[i] * inv(r[i], m[i])) % M for i in
range(n)]
    x = 0
    for i in range(n):
        x = (x + a[i] * y[i]) % M
    return x
```

5. Умножение 2 чисел по long long модулю

```
li mul(li x, li y, li mod) {
    li div = ld(x) * y / mod;
    li m = x * y – div * mod;
    return (res % m + m) % m;
}
```

6. Геометрия

```
from functools import total_ordering
from math import atan2, pi

@total_ordering
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __eq__(self, other):
        return self.x == other.x and self.y == other.y

    def __lt__(self, other):
        return self.x < other.x or self.x == other.x
and self.y < other.y

    def __repr__(self):
        return f'{self.x} {self.y}'

class Line:
    def __init__(self, a, b):
        self.a = a.y - b.y
        self.b = b.x - a.x
        self.c = -self.a * a.x - self.b * a.y

    def belongs(self, point):
        return self.side(point) == 0

    def side(self, point):
        return point.x * self.a + point.y * self.b +
self.c

    def dist(self, point):
        return abs(self.a * point.x + self.b * point.y
+ self.c) / ((self.a ** 2 + self.b ** 2) ** 0.5)

class LineCoeff(Line):
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

@total_ordering
class Vector:
    def __init__(self, a, b):
        self.x = b.x - a.x
        self.y = b.y - a.y
        self.a = a
        self.b = b

    def __mul__(self, other):
        return self.x * other.y - self.y * other.x

    def __pow__(self, other):
        return self.x * other.x + self.y * other.y

    def __lt__(self, other):
        return self * other < 0

    def __eq__(self, other):
        return self * other == 0

    def __str__(self):
        return f'{self.x} {self.y}'

    def __repr__(self):
        return f'{self.x} {self.y}'

    def polar(self):
        ans = atan2(self.y, self.x)
        if ans < 0:
            ans += 2 * pi
        return ans

    def length(self):
        return (self.x ** 2 + self.y ** 2) ** 0.5

class Segment:
    def __init__(self, a, b):
        self.line = Line(a, b)
        self.a = a
        self.b = b

    def belongs(self, point):
        if not self.line.belongs(point):
            return False
        return (min(self.a.x, self.b.x) <= point.x <=
max(self.a.x, self.b.x) and
                min(self.a.y, self.b.y) <= point.y
<= max(self.a.y, self.b.y))

    def dist(self, point):
        if Vector(self.a, self.b) ** Vector(self.a,
point) < 0:
            return dist(point, self.a)
        if Vector(self.b, self.a) ** Vector(self.b,
point) < 0:
            return dist(point, self.b)
```

```
        return self.line.dist(point)

    def seg_dist(self, other):
        if (self.line.side(other.a) *
self.line.side(other.b) < 0 and
                other.line.side(self.a) *
other.line.side(self.b) < 0):
            return 0
        return min(self.dist(other.a),
self.dist(other.b),
                   other.dist(self.a),
other.dist(self.b))

    def intersects(self, line):
        point = intersects(self.line, line)
        if point and self.belongs(point):
            return point
        return False

class Circle:
    def __init__(self, x, y, r):
        self.x = x
        self.y = y
        self.r = r

    def __repr__(self):
        return f'{self.x} {self.y} {self.r}'

    def belongs(self, p):
        return (self.x - p.x) ** 2 + (self.y - p.y) ** 2
<= self.r ** 2 + EPS

    def inside(self, circle):
        return (self.x - circle.x) ** 2 + (self.y -
circle.y) ** 2 <= (self.r - circle.r) ** 2 + EPS

def det(a, b, c, d):
    return a * d - b * c

def intersects(line1, line2):
    D = det(line1.a, line1.b, line2.a, line2.b)
    if D == 0:
        return False
    D1 = det(line1.c, line1.b, line2.c, line2.b)
    D2 = det(line1.a, line1.c, line2.a, line2.c)
    return Point(-D1 / D, -D2 / D)

def dist(point1, point2):
    return ((point2.x - point1.x) ** 2 + (point2.y -
point1.y) ** 2) ** 0.5

def cw(a, b, c):
    return Vector(b, a) * Vector(b, c) < 0

def ccw(a, b, c):
    return Vector(b, a) * Vector(b, c) > 0

def convex_hull(points):
    p = sorted(points)
    n = len(points)
    up = list()
    down = list()
    up.append(p[0])
    down.append(p[0])
    for i in range(1, n):
        if i == n - 1 or not cw(p[n - 1], p[0], p[i]):
            while len(up) >= 2 and not
ccw(up[len(up) - 2], up[len(up) - 1], p[i]):
                up.pop()
            up.append(p[i])
        if i == n - 1 or cw(p[n - 1], p[0], p[i]):
            while len(down) >= 2 and not
cw(down[len(down) - 2], down[len(down) - 1],
p[i]):
                down.pop()
            down.append(p[i])
    return down + up[1:-1][::-1]

def intersects_c_l(circle, line, dx, dy):
    sq = line.a ** 2 + line.b ** 2
    if sq == 0:
        return []
    x0 = -line.a * line.c / sq
    y0 = -line.b * line.c / sq
    d0 = abs(line.c) / (sq ** 0.5)
    if d0 > circle.r + EPS:
        return []
    if d0 > circle.r - EPS:
        return [Point(x0 + dx, y0 + dy)]
    dsq = circle.r ** 2 - (line.c ** 2) / sq
    mult = (dsq / sq) ** 0.5
    P = Point(x0 + line.b * mult + dx, y0 - line.a *
mult + dy)
    Q = Point(x0 - line.b * mult + dx, y0 + line.a *
mult + dy)
    return [P, Q]

def intersects_c_c(circle1, circle2):
    circle2.x -= circle1.x
    circle2.y -= circle1.y
```

```
    line = LineCoeff(2 * circle2.x, 2 * circle2.y,
circle2.r ** 2 - circle1.r ** 2 - circle2.x ** 2 -
circle2.y ** 2)
    ans = intersects_c_l(Circle(0, 0, circle1.r), line,
circle1.x, circle1.y)
    circle2.x += circle1.x
    circle2.y += circle1.y
    return ans
```

7. Решето Эратосфена

```
bool sieve[N + 1];
int a[N + 1];
for (int i = 2; i < N; i++) {
    if (! sieve[i]) {
        for (li j = i * 1ll * i; j < N; j += i) {
            if (! sieve[j]) {
                sieve[j] = 1;
                a[j] = i;
            }
        }
    }
}
```

Теория игр

8. Шпрага-Гранди

```
def mex(v):
    n = len(v)
    used = [False] * (n + 1)
    for x in v:
        if x <= n:
            used[x] = True
    for i in range(n + 1):
        if not used[i]:
            return i

def gr(x, used):
    trans = list()
    for i in range(1, x + 1):
        if i not in used:
            trans.append(gr(x - i, used + [i]))
    return mex(trans)
```

Функция Гранди для комбинации игр = XOR-сумма функций Гранди от каждой из игр.
XOR-сумма = 0 → выигрывает 2ой игрок
XOR-сумма != 0 → выигрывает 1ый игрок

Математические формулы

Линейность матожидания:

$M(x_1 + x_2 + \ldots + x_n) = M(x_1) + \ldots + M(x_n)$

$$S_n = \frac{b_1(1-q^n)}{1-q}.$$

$$S = \frac{b_1}{1-q}$$

$$S_n = \frac{2a_1 + d(n-1)}{2} \cdot n$$

$$1^2 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}.$$

$$\int \sqrt{a^2 - x^2}dx = \frac{x}{2}\sqrt{a^2 - x^2} + \\ + \frac{a^2}{2}\arcsin\frac{x}{a} + C$$

$$\int \sqrt{x^2 \pm a^2}dx = \frac{x}{2}\sqrt{x^2 \pm a^2} \pm \\ \pm \frac{a^2}{2}\ln|x + \sqrt{x^2 \pm a^2}| + C$$

$$\int \frac{dx}{a^2+x^2} = \frac{1}{a}\operatorname{arctg}\frac{x}{a} + C(a \neq 0)$$

$$\int \frac{dx}{a^2-x^2} = \frac{1}{2a}\ln\left|\frac{a+x}{a-x}\right| + C(a \neq 0)$$

$$\int \frac{xdx}{a^2 \pm x^2} = \pm\frac{1}{2}\ln|a^2 \pm x^2| + C$$

$$\int \frac{dx}{\sqrt{a^2-x^2}} = \arcsin\frac{x}{a} + C(a \neq 0)$$

$$\int \frac{dx}{\sqrt{x^2 \pm a^2}} = \ln|x + \sqrt{x^2 \pm a^2}| + C$$

$$\int \frac{xdx}{\sqrt{a^2 \pm x^2}} = \pm\sqrt{a^2 \pm x^2} + C$$

Площадь криволинейного сектора:

$$\mu(F) = \frac{1}{2}\int_{\alpha}^{\beta} r^2(\varphi)d\varphi.$$