

1 Contest

2 Struktury danych

3 Matma

4 Geometria

Contest (1)

sol.cpp

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

#ifdef LOCAL
auto& operator<<(auto&, pair<auto, auto>);
auto& operator<<(auto& o, auto x) {
    o << '{';
    for (int i = 0; auto y : x) o << ", " + !i++ * 2 << y;
    return o << '}';
}
auto& operator<<(auto& o, pair<auto, auto> x) {
    return o << '(' << x.first << ", " << x.second << ')';
}
void __print(auto... x) { ((cerr << ' ' << x), ...) << endl; }
#define debug(x...) cerr << "[" #x "]:", __print(x)
#else
#define debug(...) 2137
#endif

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
}
```

.vimrc

```
set nu expandtab tabstop=2 shiftwidth=2 autoindent
syntax on
colorscheme habamax
hi MatchParen ctermfg=66 ctermbg=234 cterm=underline
nnoremap ; :
nnoremap : ;
inoremap {<cr> {<cr>}<esc>O <bs><tab>
```

Makefile

```
CXXFLAGS=-std=c++20 -Wall -Wextra -Wshadow
```

```
sol: sol.cpp
    g++ $(CXXFLAGS) -fsanitize=address,undefined -g -DLOCAL \
        sol.cpp -o sol
```

```
fast: sol.cpp
    g++ $(CXXFLAGS) -O2 sol.cpp -o fast
```

test.sh

```
#!/bin/bash
```

```
for((i=1;i>0;i++)) do
    echo "$i"
```

```
1 echo "$i" | ./gen > int
diff -w <(.sol < int) <(.slow < int) || break
1 done
```

1 Struktury danych (2)

2 wavelet.cpp
Stosowanie: st – początek, ed – koniec, sst – posortowany początek.
Czas: $\mathcal{O}((n + q) \log n)$

```
struct node {
    int lo, hi;
    vector<int> s;
    node *l = 0, *r = 0;
    node(auto st, auto ed, auto sst) {
        int n = ed - st;
        lo = sst[0];
        hi = sst[n - 1] + 1;
        if (lo + 1 < hi) {
            int mid = sst[n / 2];
            if (mid == sst[0]) mid = *upper_bound(sst, sst + n, mid);
            s.reserve(n + 1);
            s.push_back(0);
            for (auto it = st; it != ed; it++) {
                s.push_back(s.back() + (*it < mid));
            }
            auto k = stable_partition(st, ed, [&](int x) {
                return x < mid;
            });
            auto sm = lower_bound(sst, sst + n, mid);
            if (k != st) l = new node(st, k, sst);
            if (k != ed) r = new node(k, ed, sm);
        }
    }
    int kth(int a, int b, int k) {
        if (lo + 1 == hi) return lo;
        int x = s[a], y = s[b];
        return k < y - x ? l->kth(x, y, k)
            : r->kth(a - x, b - y, k - (y - x));
    }
    int count(int a, int b, int k) {
        if (lo >= k) return 0;
        if (hi <= k) return b - a;
        int x = s[a], y = s[b];
        return (l ? l->count(x, y, k) : 0) +
            (r ? r->count(a - x, b - y, k) : 0);
    }
    int freq(int a, int b, int k) {
        if (k < lo || hi <= k) return 0;
        if (lo + 1 == hi) return b - a;
        int x = s[a], y = s[b];
        return (l ? l->freq(x, y, k) : 0) +
            (r ? r->freq(a - x, b - y, k) : 0);
    }
};
```

pbds.cpp

Stosowanie: s.find_by_order(k) i s.order_of_key(k).
Czas: $\mathcal{O}(\log n)$

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

```
template <typename T>
using ordered_set = tree<T, null_type, less<T>, rb_tree_tag,
```

```
tree_order_statistics_node_update>;
```

treap.cpp

Czas: $\mathcal{O}(\log n)$

```
mt19937_64 rng(2137);
struct node {
    int val, sz = 1;
    uint64_t pr;
    node *l = 0, *r = 0;
    node(int x) {
        val = x;
        pr = rng();
    }
    void pull() {
        sz = 1 + size(l) + size(r);
    }
    friend int size(node* a) {
        return a ? a->sz : 0;
    }
    friend pair<node*, node*> split(node* a, int k) {
        if (!a) return {0, 0};
        if (k <= size(a->l)) {
            auto [la, lb] = split(a->l, k);
            a->l = lb;
            a->pull();
            return {la, a};
        } else {
            auto [ra, rb] = split(a->r, k - size(a->l) - 1);
            a->r = ra;
            a->pull();
            return {a, rb};
        }
    }
    friend node* merge(node* a, node* b) {
        if (!a || !b) return a ? a : b;
        if (a->pr > b->pr) {
            a->r = merge(a->r, b);
            a->pull();
            return a;
        } else {
            b->l = merge(a, b->l);
            b->pull();
            return b;
        }
    }
};
```

Matma (3)

ntt.cpp

Stosowanie: Liczby NTT-pierwsze: $(998244353, 3) - 2^{23}$, $(754974721, 11) - 2^{24}$, $(167772161, 3) - 2^{25}$, $(469762049, 3) - 2^{26}$.
Czas: $\mathcal{O}((n + m) \log(n + m))$

```
const int ROOT = 3;
void ntt(vector<mint>& a) {
    int n = ssize(a), d = __lg(n);
    vector<mint> w(n);
    mint ww = 1, r = mint(ROOT).pow((MOD - 1) / n);
    for (int i = 0; i < n / 2; i++) {
        w[i + n / 2] = ww;
        ww *= r;
    }
    for (int i = n / 2 - 1; i > 0; i--) w[i] = w[2 * i];
    vector<int> rev(n);
```

```
for (int i = 0; i < n; i++) {
    rev[i] = (rev[i >> 1] | ((i & 1) << d)) >> 1;
    if (i < rev[i]) swap(a[i], a[rev[i]]);
}
for (int i = 1; i < n; i *= 2) {
    for (int j = 0; j < n; j += 2 * i) {
        for (int k = 0; k < i; k++) {
            mint z = w[i + k] * a[j + k + i];
            a[j + k + i] = a[j + k] - z;
            a[j + k] += z;
        }
    }
}
}
}
vector<mint> conv(vector<mint> a, vector<mint> b) {
    int n = 1, s = ssize(a) + ssize(b) - 1;
    while (n < s) n *= 2;
    a.resize(n);
    b.resize(n);
    ntt(a);
    ntt(b);
    for (int i = 0; i < n; i++) a[i] *= b[i];
    ntt(a);
    reverse(a.begin() + 1, a.end());
    a.resize(s);
    mint inv = mint(n).inv();
    for (int i = 0; i < s; i++) a[i] *= inv;
    return a;
}
}
```

Geometria (4)

tangents.cpp

Stosowanie: Wielokąt musi być CCW i $n \geq 3$. Zwraca najbliższe punkty styczne różne od a .

Czas: $\mathcal{O}(\log n)$

```
pair<pt, pt> tangents(const vector<pt>& p, pt a) {
    int n = ssize(p);
    pt t[2];
    for (int it = 0; it < 2; it++) {
        auto dir = [&](int i) {
            pt u = p[i] - a;
            pt v = p[i < n - 1 ? i + 1 : 0] - a;
            ll c = cross(u, v);
            if (c != 0) return c < 0;
            if (dot(u, v) > 0) return norm(u) > norm(v);
            return true;
        };
        auto dirx = [&](int i) { return dir(i) ^ it; };
        if (dirx(0) == 1 && dirx(n - 1) == 0) {
            t[it] = p[0];
            continue;
        }
        int s[2] = {0, n - 1};
        while (s[1] - s[0] > 2) {
            int mid = (s[0] + s[1]) / 2;
            int x = dirx(mid);
            if (dirx(s[x ^ 1]) == (x ^ 1)) {
                s[x] = mid;
            } else {
                ((cross(p[mid] - a, p[s[1]] - a) < 0) ^ it
                 ? s[x]
                 : s[x ^ 1]) = mid;
            }
        }
    }
}
```

```
    t[it] = dirx(s[0] + 1) == 0 ? p[s[0] + 2] : p[s[0] + 1];
}
return {t[0], t[1]};
}
```