Day 6 Contest Analysis, Division C

March 15, 2019

Filipp Rukhovich

Moscow Workshops ICPC Summer 2019

Problems A,B,C

These problems were solved almost all of participants.

D. Division

In this problem, given a, b, m, and you are to find such a c that $b * c \equiv_m a$ or say that there are no such a c or there are at least two such c-s.

Note that $m \leqslant 10^6$; so one could iterate over all possible c-s and check if the equivalence above is correct for each of c-s, and also calculate number of c-s satisfying the equivalence.

E. Great Theorem

In this problem, you are jist to implement a fast exponentation to calculate three degrees in $O(log\ n)$ modulo $10^9 + 7$ and then print sum (by the same modulo) of first two nombers and the last one.

F. Inverse by Prime Modulo

In multitest mode, you are to find an inverse of a by modulo $MOD = 10^9 + 9$.

The answer is a^{MOD-2} (due to little Fermat's theorem), and it can be calculated in $O(log\ MOD)$ time using fast exponentation.

G. Zeroes and Ones

It's the same problem as problem A in contest 1C; the only difference that the answer can be very big and should be calculated by modulo $MOD = 10^9 + 7$. So, instead of $dp[i] = dp[i-1] + dp[i-2]$ one should write $dp[i] = (dp[i-1] + dp[i-2])\%MOD$ storing the remainder of number of "good" sequence by MOD instead of pure such a number.

H. Prime Numbers

Given a k (in multitest mode), we are to print k-th number if sequence of prime numbers.

From sample you know that the answer will no be more then 1300000. Then, one can just find all the prime number from 2 to 1300000, store them in a vector/array and then find k-th prime number for any given k in constant time.

I. Factorization

Given n, you are to find its all the prime positive integer divisors with their degrees in factorization.

The solution with $O(\sqrt{(}n))$ complexity follows:

```
for (int i = 2; i*i <= n; ++i)
    while (n % i == 0) {
        n /= i;
        cout << i << ' ';
    }
if (n > 1)
    cout << n << ' ';
```

J. Coprimes

Given an integer n, you are to find a $\phi(n)$.

Find a factorization of n from problem *I*. Let it be $n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}$. After that, just calculate $\phi(n)$ as $(p_1 - 1) * p_1^{a_1 - 1} * (p_2 - 1) * p_2^{a_2 - 1} * \dots * (p_k - 1) * p_k^{a_k - 1}$.

The complexity of the solution is $O(\sqrt{(n)})$ per test.

K. C_n^k

Given n, m, l, we are to find $\sum_{k=1}^{l} C_n^{mk}$ by prime modulo $P = 10^9 + 7$.

First of all we know that $C_n^k = \frac{n!}{k!(n-k)!}$ is an integer value. Then if we find $n!$, $k!$ and $(n-k)!$ modulo P, one can calculate C_n^k in $O(log\ P)$ time using modulo inversing (as in problem F) by modulo P.

Let $fact[0..n]$ be an array such that $fact[i] = i!\%P$, $i = 0, 1, 2, \ldots, n$. It can be done in $O(n)$ using the fact that $fact[0] = 1$, $fact[i] = (fact[i-1] * i)\%P$, $i = 1, 2, \ldots, n$. After that, just calculate the answer calculating each C_n^{mk} in $O(log\ P)$. So, the total complexity of the solution is $O(n + l\ log\ P)$.

It also can be done in $O(n\ log\ P + l)$ if one precalculates inverses of all $fact[i]$-s by modulo P.

L. GCD

In this problem, your are just to find GCD of two numbers using Euclid's algorithm.

M. Diophantine Equation

Given $A, B, C \in \mathbb{Z}$, you are to find an integer solution of equation $Ax + By = C$ with as minimal nonnegative x as possible.

First of all, if C is not a multiple of $GCD(A, B)$ then the equation doesn't have any solution. Otherwise, write a recursive procedure $findAns(A, B, C)$ which gets integer A, B, C and returns a solution (x, y) of equation which x is as minimal nonnegative as possible. But how does it work?

If $B = 0$ then x should be C/A, and y can be any number, for example 0.

Otherwise, let q, r be a quotient and remainder of division of A by B. Run $findAns(B, r, C)$, and let (x_1, y_1) be a result of this run. It means that $C = B * x_1 + r * y_1 = B * x_1 + (A - B * q) * y_1 = A * y_1 + B * (x_1 - q * y_1)$. Then, $(x_0, y_0) = (y_1, x_1 - q * y_1)$ is some solution of the initial equation. From the theory we can see that solutions of this equation can be written as $(x_0 - Bt, y_0 + At)$, $t \in \mathbb{Z}$. Then, the optimal x is $rem_B(x_0)$; calculate this x and corresponding y as $(C - Ax)/B$ and return it as the answer.

The complexity of the solution is $O(log(A + B))$ per one test.

N. Billiard Table

There is the only ball at the point A on the rectangular $x \times y$ billiard table. The ball is hit in such a way that before first meeting the border, the ball meets the point B. It is needed to check whether the ball will move, possibly after some number of collisions with border of billiard, through point C or not. The ball is considered to be perfectly elastic, and linear sizes of the ball and frictional force are negligible. During the collision of the ball with the border, angle of incidence is equal to angle of reflection.

N. Billiard Table

We will use unfolding method. More precisely, when the ball meets a border, we will not reflect the ball — instead of that, we will reflect a table with point C with respect to the border we collide. In such an interpretation, trajectory of the ball is a straight ray AB, and we are to find out whether this ray contains at least one point from set S_C or not; here $S_C = S_1 \cup S_2 \cup S_3 \cup S_4$, for:

- S_1 as $\{(x_c + 2xk, y_c + 2yl) \mid k, l \in \mathbb{Z}\}$;
- S_2 as $\{(x - x_c + 2xk, y_c + 2yl) \mid k, l \in \mathbb{Z}\}$;
- S_3 as $\{(x_c + 2xk, y - y_c + 2yl) \mid k, l \in \mathbb{Z}\}$;
- S_4 as $\{(x - x_c + 2xk, y - y_c + 2yl) \mid k, l \in \mathbb{Z}\}$;

We will describe how to solve the problem for S_1; for other sets, the solution is the same.

N. Billiard Table

First of all, one can see that if some point $p \in S_1$ lie on the line containing the ray AB, then point $p' = p + xy * \vec{AB} \in S_1$ will lie on the same line too. It means than we can consider the whole line, not only the ray.

Then, some points lie into line AB if and only if $[B - A, p_{kl} - A] = 0$ for some $k, l \in \mathbb{Z}$, $p_{kl} = (x_c + 2xk, y_c + 2yl)$. If you the formula of cross product, the will get the following diophantine equation with respect to k, l:

$$(x_b - x_a) * (y_c - y_a + 2yl) - (x_c - x_a + 2xk) * (y_b - y_a) = 0.$$

We don't need to solve it — we are just to find out whether the solution exist. To do that it's enough to bring the equation into interpretation $Ax + By = C$ for some integer A, B, C and then print "YES" if C is a multiple of $GCD(A, B)$ and "NO" otherwise. The complexity of solution is $O(log(x + y))$.