Criterion 2020 Round 1

January 31, 2020 - 3pm +0600

FURQAN SOFTWARE

ARDENT PROGRAMMERS

Problemset Analysis

SETTERS

Dhruba Mitra
Md. Hasinur Rahman
Md. Moshiur Rahman
Risal Shahriar Shefin
Shahwat Hasnaine
Tahsin Masrur
Tauhedul Islam Tanu

Problem A: Welcome to Criterion Rounds

Setter: Md Hasinur Rahman

It is easy to prove that for any number n and bases > sqrt(n) there will be no representation where the number will contain 0 at its second least significant digit. So it is enough to check for bases <= sqrt(n).

As there are 10^6 test cases you have to precalculate the results to avoid TLE.

Time complexity: **O(n*sqrt(n) + Q)** where **Q** is the number of queries and **n** is the highest possible number in the query.

Problem B: A Journey Towards One

Setter: Md Moshiur Rahman

If you analyze, you'll find it is better to divide N by 2 when available.

So,

dp[n] = 1 + dp[n / 2], when N is even.

When N is odd, we have to increase/decrease N by 1 and then divide it by 2.

dp[n] = 2 + min(dp[(n - 1)/2], dp[(n + 1)/2])

This can be solved easily with a recursive solution with the base case, if N = 1, then the answer is 0.

Problem C: Magic Moments

Setter: Dhruba Mitra

Basically, you have to take the largest *k* numbers from the *n* numbers given in input and print the minimum number among them which is actually the *k*-th maximum number among the *n* numbers. This can be easily done by sorting. It was the giveaway problem of this contest.

Problem D: Straightforward

Setter: Tauhedul Islam Tanu

At first, replace the given bracket sequence by an integer array a.

In this array,

a[i]=(count of opening bracket till index i - count of closing bracket till index i);

Now in each query, we need to find the maximum possible index x, in the range from l to r where a[x]=a[l-1] and no element from a[l] to a[x] is less than a[l-1]. To do this, first check if there is any element from a[l] to a[r] that is less than a[l-1]. Find the first occurrence of such an element. This can be done using segment tree in O(NlogN) time complexity.If such an element exists, replace r with the index of that element. Then find the maximum index y between l and r where a[y]=a[l-1]. This can be found by using binary search . The output will be y-l+1.
Time complexity O(NlogN).

This problem can also be solved by using sparse table. Time complexity is same O(NlogN).

Problem E: Team Forming Editorial

Setter: Risal Shahriar Shefin

Editorial: Observing the constraints, it's pretty obvious that approximate O(nlogn) solution is needed to pass. So, we need a data structure that can join two disjoint sets and measure the current size of any set efficiently. Disjoint Set Union (DSU) is a data structure that can perform these efficiently. You can read about DSU here: https://cp-algorithms.com/data_structures/disjoint_set_union.html. Type 1 and Type 2 queries are pretty straightforward operations in DSU actually. In type 1 queries, we will store the assignment operations in a vector or an array.

Let's focus on type 3 queries. Student u and a special number x are given. We will iterate from the last valid assignment operations to the first valid assignment operations and roll back the updates that occurred in the ith assignment operation. While iterating and rolling back, if the size of the team of student u becomes less than x in an operation, that means this operation is the special operation and we will stop iterating. As we have rolled back the updates of this special operation already, we have to execute this operation again. The later valid assignment operations after this special operation are already rolled back. So, they have no effect on our DSU structure now. Don't forget to erase those later

operations from the vector. Erasing can be done in O(n). Now, we can print the size of the student u's team.
Why this will fit in time limit? Look, we are rolling back the updates of a canceled assignment operation only once. After that, we are erasing them from the vectors. In each iteration, we are finding the size of the student u's team which can be performed in O(logn). So, the whole iteration, rolling back and checking size procedures have O(nlogn) complexity.

How to roll back?

We will manage a rank-based DSU structure. So, the height of any tree in our DSU structure won't be greater than O(logn). As a result, the complexity of "find parent" operation in our DSU won't be greater than O(logn) even if we don't do path-compression. To make things simpler, we are actually avoiding path-compression here :D . If we manage DSU like this, then in an assignment operation, only these updates are occurring:
1. Parent of a node(student) changes
2. Rank of a set(team) might change
3. Size of a set(team) changes
We will manage a stack keeping track of these updates in each assignment operation. While iterating, we will just get the information on changes from the stack and restore the changes in our DSU structure. Each restoration can be done in O(1) complexity. Rolling back an assignment operation is basically disjointing the set(team) of student u and the set(team) of student v.

Judge Solution: https://paste.ubuntu.com/p/KQKVnkDTr8/
Overall Complexity: O(nlogn)

Problem F: I am Good

Setter: Tahsin Masrur
Time Complexity: $O(\sqrt{px}\log_2(px))$

Let L be the length of X (number of digits).

Then,

$$f(X,K)=\sum_{i=0}^{K-1}10^{iL}X = X\sum_{i=0}^{K-1}10^{iL} = X\frac{10^{KL}-1}{10^L-1}$$

Now, we need to find the minimum K such that the above expression is divisible by P. That means $\frac{X(10^{KL}-1)}{P(10^L-1)}$ needs to be an integer.

Let G be the GCD of X and $P(10^L-1)$.

With prime factorization, we can easily find

$$P' = \frac{P(10^L - 1)}{G}$$

At this point, we basically needs to find whether $\frac{10^{KL} - 1}{P'}$ is an integer i.e. we need to find the minimum K such that

$$10^{KL} - 1 \equiv 0 \pmod P)'$$

$$10^{KL} \equiv 1 \pmod P)'$$

Now, let’s try to solve a simpler problem - finding the minimum value of Z such that $10^Z \equiv 1 \pmod N)$

When 10 and N are not coprime, it can be proven that Z can not exist.

Let's take a look at Euler's theorem now. It states that if n and a are coprime positive integers, then

$$a^{\varphi(n)} \equiv 1 \pmod n)$$

So, for $10^Z \equiv 1 \pmod N)$, one possible solution is $Z = \varphi(N)$. But can't a smaller value for Z exist? Yet, it can. But even if it does, it must be a divisor of $\varphi(N)$. The proof of that is being left for the reader to figure out.

Let's return to our original equation to solve, $10^{KL} \equiv 1 \pmod P)'$.

Within the constraints of this problem, we can simply traverse on all the divisors D of $\varphi(P')$ and find the optimal K for that divisor if $10^D \equiv 1 \pmod P)'$. Needless to say, we choose the minimum value of K as output.