Solutions Gran Premio de México 2019 - Primera Fecha

Club de Algoritmia ESCOM

May 5th, 2019

This document contains the solutions to the problems of the first stage of the "Gran Premio de México 2019".

The proposed solutions for these problems have been developed by the "Club de Algoritmia ESCOM" and are not the official solutions. This document is intended to help people to improve in competitive programming and to continue upsolving the problems.

Development team

The following persons helped to develop this document. Persons in this list could be from another university or another competitive programming community other than "Club de Algoritmia ESCOM".

Filiberto Fuentes Hernández, ESCOM IPN
Abraham Omar Macías Márquez, ESCOM IPN
Bryan Enrique González Vélez, ESCOM IPN

Problem A - Add and substract

Given N numbers a_i you need to compute the following sum:

$\sum_{i=1}^{N} \sum_{j=i+1}^{N} (a_j - a_i)$ if and only if $|a_i - a_j| > 1$

So, taking in consideration only the elements that holds the condition, we need to compute the following sum:

$\sum_{i=1}^{N} (\sum_{j=i+1}^{N} a_j) - (N - i) * a_i$

To solve this problem, we can use an array C to maintain the number of occurrences of each number as the numbers are not very big. Then, just iterate from the first element to the last one keeping the sum of all of the elements that are in front of the number being processed.

To compute the answer for the current element we apply the following expression and update the sum by subtracting the current number:

$answer = (sumInFront - C[a_i - 1] * (a_i - 1) - C[a_i] * a_i - C[a_i + 1] * (a_i + 1))$
$- (totalElementsInFront - C[a_i - 1] - C[a_i] - C[a_i + 1]) * a_i$

Then, just add all the answers and print it.

Dificulty: Very easy

Expected Complexity: O(n)

Problem B - Box delivery

Prerequisites: Basic combinatorics, Modular arithmetic

Jaime needs to deliver K boxes on N days. In some of these days he must deliver at least 1 or 2 boxes depending on whether the supervisor of the supervisor's boss will be in the new branch. On other days he can deliver any number of boxes.

So, lets consider the days he must deliver either 1 or 2 boxes. We will make sure he delivers those boxes on all of these days by subtracting those boxes from the original K boxes he needed to deliver.

Once we make sure he holds the conditions by subtracting the boxes he must deliver in the mentioned days, the problem we need to solve is the following:

Given K boxes and N days, in how many ways we can distribute the boxes such that all the boxes are delivered on any of the N days.

This problem can be solved by the following binomial coefficient:

$$\binom{K+N-1}{N-1}$$

This problem is pretty well known in combinatorics and it's named as the *Stars and bars problem.*

Dificulty: Easy

Expected complexity: O(n $\log_2$ n)

Problem C - Connecting cities

Prerequisites: Minimum Spanning Tree

Given a set of roads and a proposal of new roads to be used in the city, you need to print the minimum cost to make the cities connected.

This is a classic problem where we need to get the minimum spanning tree or better known as MST.

Create the graph with the old roads without considering the costs of them. Then, with the new edges run any MST algorithm and print the answer. You can use either Kruskal's algorithm of Prim's algorithm.

There are several cases to care about:

- If the cities were already connected with the old roads, then print "Thank you, Goodbye".
- If the cities were not connected after running MST algorithm, print "You better hire someone else".
- Otherwise print the minimum cost.

Dificulty: Easy

Expected complexity: O(n $\log_2$ n)

Problem D - Determining rally paths

Prerequisites: Lowest Common Ancestor, Modular Multiplicative Inverse

Given a tree with N vertices numbered from 1 to N and a label C_i in each node i and Q pairs of values (a, b), we have to concatenate the labels in the path from a to b and print the remainder of that number after dividing it by 188888881.

The main idea is to split the answer in two different paths: one representing the path from a to $LCA(a, b)$ and the other one the path from $LCA(a, b)$ to b.

Lets say $lca = LCA(a, b)$ is the lowest common ancestor between a and b.

Also $lcaParent$ is the parent of the node lca in the tree.

First path we are going to get is the one from lca to b. If we run a dfs or bfs from the node zero, that is going to be our root, then we can store the number formed from the root to each node of the tree.

Once we have this number formed in each of the nodes, we can easily compute the number formed from lca to b with the following formula:

$lcaTob = numberFormed_b - (numberFormed_{lcaParent} * 10^{len_b - len_{lcaParent}})$
where len_u is the length of the number formed at node u.

With this, we have successfully solved the problem of the path from lca to b. But now we need to get the second path and merge it with the one we have already computed.

To get the answer of the path from a to lca, we can compute another value in our dfs or bfs. This value will be the number formed from each node of the tree to the root. That means in the reverse number formed.

Once we have the second value at each node, we can use the following formula to get the answer from a to the node lca (Without considering the value of the lca node):

$aToLca = \frac{reverseNumberFormed_a - reverseNumberFormed_{lca}}{10^{len_{lca}}}$

Also, as the problem asks us to print the answer modulus 188888881 (and because we can not store the whole number of a path because it could be huge) we can not make the division just like that.

We need to compute the multiplicative inverse of the dividend and the divisor and here we can use Fermat's little theorem or the extended euclidean

algorithm, it is up to you. With that said, then our formula ends as follows:

$aToLca = reverseNumberFormed_a - reverseNumberFormed_{lca} * 10^{-len_{lca}}$ where the power of 10 is the modular inverse of $10^{len_{lca}}$ and the number 188888881.

Once we have these 2 disjoints paths and their answers, we can merge them together by applying the next formula:

$answer = aToLca * 10^{len_b - len_{lcaParent}} + lcaTob$

To get the *lca* we can use either binary lifting or Euler tour and a data structure such as segment tree.

Read carefully at each step to understand completely the solution as it could be a little bit confusing if you are new to modular arithmetic or to the lowest common ancestor problem. Read the prerequisites to know which topics you should learn in order to solve this problem.

Dificulty: Medium

Expected complexity: O(n $\log_2$ n)

Problem E - Egyptian binary system

This is a very easy problem. Given a string you need to print how many substring represents an odd number in its binary representation with no leading zeroes.

To solve this problem for each suffix that starts with a digit 1, add the number of ones in the $i_t h$ suffix to the answer.

Dificulty: Very easy

Expected complexity: O(n)

Problem F - Forecasting rock-paper-scissors

Prerequisites: Union-Find

Note that if we construct a graph with n nodes and m edges, where an edge connect the nodes x and y if the result of the match between the players x and y is given in the input, then the result between two players can be predicted if and only if they are in the same connected component.

Let's assign an integer $f(a)$ to the players in a component, such that $f(x) - f(y) \equiv 0 \mod 3$ if the match between players x and y finishes in a draw, $f(x) - f(y) \equiv 1 \mod 3$ if player x wins, and $f(x) - f(y) \equiv -1 \mod 3$ if player y wins.

Maintain an Union-Find Data Structure with rank. Initially $f(a) = 0$. To join two nodes x and y, we need update the values in the set of y such that $f(x) - f(y) \equiv z \mod 3$. Let's assume that $rank(set(x)) \geq rank(set(y))$ (if not, just swap x and y and multiply z by -1), then add $f(x) - z - f(y)$ to all the nodes from the set of y. To update optimally, maintain $lazy(set(y)) = f(x) - z - f(y)$. Then, to obtain the value $f(a)$ of a node a, just go through the path from a to the root of its set, and add the values $lazy(b)$ of all the nodes b in this path. Also store a value $day(set(y)) = k$, that means that we connected the node $set(y)$ with its parent in the $k-th$ day. Note that you **must not** do path compression.

To answer a query, if the nodes x and y aren't in the same set, then print -1. Otherwise, obtain the values $f(x)$ and $f(y)$, if $(f(x) - f(y)) \equiv 1 \mod 3$ then x wins, if $(f(x) - f(y)) \equiv -1 \mod 3$ then y wins, and otherwise there is a tie. Finally, to compute the earliest day in the tournament in which the match could be predicted, just obtain the maximum value for $time(a)$ in the path from x to y, except for the LCA of x and y.

Dificulty: Medium

Expected complexity: O(n $\log_2$ n)

Problem G - Going to the world finals again

Prerequisites: Pollard's rho algorithm, Number Theory

This problem asks us for the total of numbers d such that if we start adding $d, d+1, d+2, ...$ we will suddenly get X as the sum.

Lets express our problem with a number d and N. N means the number of times we will add each term to the sum. So, if we have d and N, we will add the following:

$d + (d+1) + (d+2) + ... + (d+N-1) = X$, with $d > 0, N > 1$

Playing a little bit with this expression we can get the following:

$\sum_{i=0}^{N-1}(d+i) = X$

$\sum_{i=0}^{N-1} d + \sum_{i=0}^{N-1} i = X$

$N * d + \sum_{i=1}^{N-1} i = X$

$N * d + N * (N-1)/2 = X$, multiplying by 2 both sides

$N * d * 2 + N * (N-1) = X * 2$, factoring N

$N * (d * 2 + N - 1) = X * 2$

We know that N and d are integers, and from the expression above we notice that $X * 2$ must be divisible by N.

Finally, we just have to fix N to every divisor from $X*2$, calculate d and add one to the answer if d is integer and bigger than 0 and the sum of N consecutive numbers starting from d is equal to X. Remember to use Pollard rho algorithm for finding the divisors, since X could be as big as 10^{15}.

Also this problem can be solved by getting the odd divisors of X greater than 1. For a more detailed explanation of why this is true you can check the following links:

- Sums of consecutive integers.
- On the representation of the integers as a difference of nonconsecutive triangular numbers.

Dificulty: Medium

Expected complexity: O($\sqrt[4]{X * 2}$)

Problem H - Husbands association

This is a simulation problem.

Sort the elements by decreasing order and simulate the first approach.

Then, reverse the elements and apply the second approach.

Dificulty: Very easy

Expected complexity: O(n $\log_2$ n)

Problem I - Inspecting PIN Numbers

Prerequisites: Modular Arithmetic, Fenwick Tree

Let's build an array $A[1...10^5]$, where $A[k] = \frac{k!}{10^{Z_k}} \% 10^5$ and Z_k is the number of trailing zeros in $k!$. In other words, $A[k]$ stores the PIN number from $k!$. To answer a query, we need to count the numbers between x and y in the subarray $A[l...r]$.

Consider the prime factorization if $k!$: $2^{b_1} * 3^{b_2} * 5^{b_3}$.... The number of trailing zeros in $k!$ equals to $min(b_1, b_3)$. Since 10^{Z_k} may not be coprime with 10^5, it doesn't have modular inverse, so we need to split $k!$ in its two and five factors, and in its other factors. To build A, consider $A'[k] = A'[k-1] * k'$, where k' is equals to k divided by its two and five factors. Then, $A[k] = A'[k] * 2^{b_1 - min(b_1,b_3)} * 5^{b_3 - min(b_1,b_3)}$.

Now, to answer the queries, we can use a Persistent Data Structure, Wavelet Tree, SQRT decomposition, etc. Let's see an offline approach with Fenwick Tree. Add an event in $l-1$ and r for each query. Iterate the array from 1 to 10^5, in the $i-th$ iteration sum 1 in the $A[i]-th$ position of the Fenwick Tree. To calculate the queries, add $-sum(x, y)$ in the $(l-1)-th$ iteration and add $sum(x, y)$ in the $r-th$ iteration to the answer of the query.

Dificulty: Medium

Expected complexity: O(n $\log_2$ n)

Problem J - Jaimina party invitations

Prerequisites: Modular arithmetic, Binary Exponentiation, Euler's totient function.

Given a rectangular sheet of size N x M we need to divide a sheet of paper by making cuts satisfying the following conditions:

- The cuts must be parallel to the sides of the sheets.
- All the resulting rectangles must have the same dimensions.
- The number of generated rectangles must be p^q.

If we could generate all of the possible ways of cutting the sheet of paper into p^q rectangles, what will be the sum of the perimeters of all the generated rectangles?

To solve this problem we should focus on the number of rectangles that we need to generate from a sheet of paper. As we need to generate p^q rectangles, we know that in order to do that we could make H horizontal cuts and V verticals cuts with the same distance between them to satisfy the three conditions.

One more thing to see is that if we make H horizontal cuts and V vertical cuts we will generate $(H+1)*(V+1)$ rectangles.

As we need to have p^q rectangles, then we need the following:

$p^q = (H+1)$ * $(V+1)$ we can see that $H+1$ must divide p^q and the same for $V+1$.

So, all the possible configurations for $(H+1)$ are all the divisors of p^q.

Once we get the divisors of p^q and play a little bit with some formula to get the perimeter, the answer is the following:

$answer = sumOfDivisors * 2 * (N+M)$

To get the sum of divisors we can use Euler's totient function to compute it and as the number is a prime elevated to some power, then the formula is very easy to code.

Dificulty: Medium

Expected complexity: O($\log_2$ q)