Editorial
of
New Year Celebration Contest 2017

Contest Link: https://algo.codemarshal.org/contests/new-year-2017

Event Link: https://www.facebook.com/events/1738093939842780/

Arranged by

প্রোগ্রামিং প্রবলেম (Programming Problem in Bengali)

Problem A: Awesome Childhood

Setter: **Hasan Abdullah**

Analysis: Answer is G/2. G is no more than 2^32. To store 2^32 you need to use LONG LONG INT.

Complexity : O(1)

Problem B: Sum of a sequence

Setter: **Mohammad Sheikh Ghazanfar**
Alternate Writer: **Md Imrul Hassan**

Analysis: Sum of the sequence is easy to be in any base b, if we know the i-th number. We can see that the :
1st term of the sequence is : b^0
2nd term of the sequence is : b^1
3rd term of the sequence is : $b^2 + b^0$
4th term of the sequence is : $b^3 + b^1$
.. .. ..
.. .. ..
Thus, the 2*n-th term is : $b^{(2n-1)} + b^{(2n-3)} + b^{(2n-5)} + ... + b^1$
And 2*n+1 th term is : $b^{(2n)} + b^{(2n-2)} + b^{(2n-4)} .. .. .. + b^0$
From here we can see that odd and even both term can be made by summing the geometric series stated above. Thus any term can be made and be merged to create the first n terms. Which themselves create a geometric series again. Sum of the geometric progression is expected in the problem. For better (and brief) understanding -**http://tinyurl.com/gwg2d26.**

Complexity : O(logn).

Problem C: Jerry And His Cheese Chase

Setter: **Md Imrul Hassan**
Alternate Writer: **Mohammad Sheikh Ghazanfar**

Analysis:
This is trial based simulation problem. The input constraint is low enough that an efficient naive solution will pass the time limit.

We need to try each starting position for Jerry and calculate the minimum moves required to eat C cheeses. However, there is no advantage of starting from a cell which does not have cheese in it. So, we have to to try at most K cells as possible starting position.

As a data structure, we can store the position as (row, column) pair for each position where there is cheese. We also need to store whether a cell is visited or not while computing the path.

Next part is just simulating the movement of Jerry. From each cell with cheese, we have to find next cell that he will travel. For this we can just check all cells with cheese not already visited and find the closest cell. Finally, we stop Jerry eats C cheeses and continue with the next starting position.

The solution is simple, but here few things to make unnecessarily completed. There is no need for complicated data structure other than vector of pairs. No sorting is needed as if you store the pairs in the order you read in, the cells are already sorted from top to bottom and left to right. Then find the minimum distance cell with a single loop by updating the minimum cell whenever you find a cell with closer distance. Also, you don't need to store the closest cells from each cell using priority queue either. Pre calculating distance between each pair of cells can help make the solution a bit faster, but is not required.

Complexity : O(K^2 * C).

Problem D: Pile Game

Setter: **Mohammad Sheikh Ghazanfar**
Alternate Writer: **Md Imrul Hassan**

Analysis: It's easy to understand that the whole concept is built on NIM game. But the problem is now to find who is winner but the number arrangements Bob can win. There is no other way except checking all the possible ways with dp on three terms (number of piles we are working on, number of stones left, xor of arrangements done till now). It might be noted that there is a inner loop to fix the number of stones that being selected the specific pile. Some didn't pass the Time Limit because they tried in a bottom up technique and rebuilt the whole table for each case. Thus there complexity is O(T*n*k*128*k) [Here T is number of test cases and 128 is the highest possible xor value]. In spite we can Precalculate the whole 5050 possible test cases in O(n*k*128*k) and then answer them in O(1) for each test case.

[This problem also cause Runtime Error (RE) for a few because of setting the highest possible xor term as 128.]

Complexity : O(n*k*128*k) [For all test cases].

⇒ $O(n^4)$ [As the inner loop didn't go to it's limit n always thus on average it's pretty less than n^4 and a little greater than n^3]

Problem E: Xor magic

Setter: **Moshiur Rahman**
Alternate Writer: **Mohammad Sheikh Ghazanfar**

Analysis:
Statement was intentionally made difficult to understand . The simpler version of the statement could be,

“You are given N non negative Integer numbers. Your have to make XOR-sum of the N numbers equal to zero by replacing only one number. Your task is to pick such a number for which abs(given_number - replaced_number) is the minimum.”

Also notice that upper limit of those integers **won’t fit into 32 bit unsigned integer.**

Solution idea:
From Xor property we know X^X=0 and 0^X=X . Here , ^ symbol refers bitwise-exclusive-or (xor) operation.
Lets denote n integers as, A1,A2,....., An
calculate, S = A1^A2^.....^An
Now, answer = min(abs((S^Ai)-Ai)) for each Ai (0<i<N)

Complexity : O(n)

Problem F: Moumita and Assignments

Setter: Parvez Muntasir Supto
Alternate Writer: Md Imrul Hassan

Analysis:

Let the array elements are $A_1, A_2, A_3, ..., A_n$. Expanding the formula for F(A) we get,

$F(A) = (A_1-A_1)^2 + (A_1-A_2)^2 + ... + (A_1-A_n)^2$
$+ \quad (A_2-A_2)^2 + (A_2-A_3)^2 + ... + (A_2-A_n)^2$
$+ \quad ... +$
$+ \quad (A_{n-1}-A_{n-1})^2 + (A_{n-1}-A_n)^2+$
$+ \quad (A_n-A_n)^2$
$= (n-1)(A_1^2 + A_2^2 + A_3^2 + ... + A_n^2)$
$- \quad (2A_1A_2 + 2A_1A_3 + ... + 2A_1A_n + 2A_2A_3 + 2A_2A_4 + ... + 2A_2A_n + 2A_3A_4 + 2A_3A_5 + ... + 2A_3A_n + ... + 2A_{n-1}A_n)$ ..... (1)

We know,

$(A_1+A_2+ .... + A_n)^2 = A_1^2 + A_2^2 + A_3^2 + ... + A_n^2$
$+ \quad (2A_1A_2 + 2A_1A_3 + ... + 2A_1A_n + 2A_2A_3 + 2A_2A_4 + ... + 2A_2A_n + 2A_3A_4 + 2A_3A_5 + ... + 2A_3A_n + ... + 2A_{n-1}A_n)$ .....(2)

From (2), (1) can be rewritten as,

$F(A) = (n-1)(A_1^2 + A_2^2 + A_3^2 + ... + A_n^2)$
$- \quad ((A_1+A_2+ .... + A_n)^2 - A_1^2 + A_2^2 + A_3^2 + ... + A_n^2)$
$= n*(A_1^2 + A_2^2 + A_3^2 + ... + A_n^2) - (A_1+A_2+ .... + A_n)^2$ ..... (3)

Using (3) each query can be done in O(1). But there is a catch. The mod value is very large, so normal long long multiplication will result overflow. To handle this, you can use Russian Peasant Method [1] or use __int128 [2].

[1] http://www.cut-the-knot.org/Curriculum/Algebra/PeasantMultiplication.shtml
[2] https://gcc.gnu.org/onlinedocs/gcc-4.8.3/gcc/_005f_005fint128.htm

Complexity: Using Russian Peasant Method for multiplication each query will take O(m), m = number of bits in the number. So overall complexity is O(q*m) for each case.

Problem G: Coin Change

Setter: **Mohammad Sheikh Ghazanfar** and **Md Imrul Hassan**

Analysis : This is nothing but a variant of popular coin change problem. But in spite of adding the coins what Alice have, Bob can also return back some coins to balance the amount asked from Alice. The whole problem can vulnerabil with the constraint $(2 + N + M <= 20)$. So, we generate all possible value(X) that Alice can give (by just adding his own coins or by taking back some from Bob). It will cost $0(2^{(n+m)})$. It will need the same amount of address to store all of them. Finally in you can sort and do Binary Search or you insert them into a set and then try to find if the value is there of not. Both of them will cause same amount of complexity.

[N.B. : You don't have to separately maintain what Alice and Bob can give. As giving someone an amount is opposite of taking the amount from someone. Say we can search the value (X) to see if that can be made or not and for Bob can search (-X). That will make you code faster.

Complexity : Generating all the possible values+inserting them in a set / sorting them / mapping them + process each query

$$\Rightarrow$$

$$O(2^{(m+n)} + (m+n) * 2^{(m+n)} + (q * log(2^{(m+n)})) = O(2^{(m+n)} + (m+n) * 2^{(m+n)} + (q * (m+n))$$

Further Challenge : What if $2 <= N + M <= 32$. Obviously the above explanation will not work then. Can you manage to solve it then ? To make your life easier here is the I/O to accomplish this challenge . (https://drive.google.com/open?id=0B0_dCKKIjeFSclpIWHVROVN0dkU)

Problem H: Find The Treasure

Setter: **Al Mamun**
Alternate Writer: **Mohammad Sheikh Ghazanfar**

Analysis: “The optimal path is the shortest path. Because in the shortest path Robo has to give the least amount of gold coin.” So we can conclude that the whole problem is based on finding the distance of the shortest path from the destination. Let it’s x. So the profit Robo can made = Gold coins in the destination node - x
Obviously Robo **won't** travel if his profit is negative and Robo **can’t** travel if there is no path.

Complexity : O(n+m).

-- Thank You --