IGNITERS HACKATHON - CP SUBMISSION
YASHVANTH S
CS24I1029

Qn 1: The Enchanted Array

My Approach:

By first reading the question, i thought of the brute-force solution which is creating a nested for loop which would have a time complexity of O(n.q) and inside them, an if statement to check whether the number to be enchanted or not. But given that, n and q would approach 10^5 each and together it exceeds 10^9. So, i found that the brute force solution wont work here. So i read the question again to get a hint and i found that value of q would always be less than 30. So instead of taking till 10^5 iterations for q i can store the frequency of values of q in an c++ stl vector structure of size 31. But it didnt work as the order of execution will affect the number of enchantments. So i was getting wrong outputs. Final optimized solution was the brute force only in this problem and the output is verified.

Time Complexity Analysis:

O(n x q) per test case

- For each exponent, we loop over n elements once.
- So per test case: ≈ O(n x q).

Space Complexity Analysis:

We store the initial array in (arr) vector and exponents in (exp) vector and a vector of vector (res).

So, the space complexity is ≈ O(n+q)

Qn 2. Bitland's Mystic Subarray

My Approach:

So, basically we are given an array of n elements, we are supposed to find all possible continuous subarray and perform bitwise AND on them and maximum among them is the Bitland's maximum mystic power. I took a nested for loop to calculate all possible continuous subarray as well as max power. As you add more numbers to your

AND, the result wont become larger, it can only stay the same or get smaller. This brute force is the solution i came up with and i guess this is moderately optimized and the output is supposed to be 3 and 2 but given 0 and 2.

Time Complexity Analysis:

Outer loop runs for n times and inner loop run up to n times

So, the overall time complexity is O(n^2) .

Space Complexity Analysis:

I'm using a result array here to store the max power. The size of the array is t(no. of test cases). So, in my code, the space complexity is O(t).

But, it is optional, the space complexity can be reduced to O(1) if we avoid the res array and print as soon as you get the output.

So, the space complexity is O(1).

Qn 3 The Prime Guardian's Test

My Approach:

According to my knowledge, the first idea that came in my mind is the brute force one, that is getting the next prime p1 and p2 by giving d and p1+d as input respectively and To check prime i used a for loop till sqrt(n). I couldn't find more optimized solution limited to my knowledge.

Time Complexity Analysis:

Every time I call the function to check prime, it takes O(sqrt(n)) to give output.

So for each test case: O(sqrt(n)) (around 300 iterations)

Space Complexity Analysis:

In my code, Major space is taken by res array to store smallest key on each test case, which is O(t). This can be reduced to O(1) if we dont use the res array, just by directly printing as soon as the output is received.

So, overall: O(1)

Qn 4 The Twin Monuments

My Approach:
According to my understanding, we are supposed to find the no. of grid cells(the point coordinates) from which the two monuments can be visited in a L-shape move (given a,b -> the L move). We can achieve it by selecting 8 possible cells each for the two monuments and find the intersection of them and those will be the points from which the scouts can reach both monuments simultaneously with their L move.
So thats the idea i got to solve this question.

Time Complexity Analysis:
I have used 2 for loops in my code, one for generating the possible cells for each monuments and one for counting the intersection. Actually both takes O(8) (constant time).

So, overall time complexity is O(t).

Space Complexity Analysis:
The auxiliary space i used is three vectors and one set as inserting max of 8 elements. So, constant extra space used.

Overall space complexity is O(1).