Date _____ Page _____

Day - 194

Sliding window - 2

* **Subarray Sums Divisible by k.**

3	5	6	3	9	4	6	9

k = 7

no. of

⇉ We have to return subarray sums divisible by k.

⇉ In brute force approach, we can find all the subarrays sum and then check it is divisible by k or not.

3	5	6	3	9	4	6	9

3 8 14 17 26 30 36 45

⇉ We can use prefix sum.

⇉ So, we can see that 14 is divisible by 7, so, it's our one of the answer.

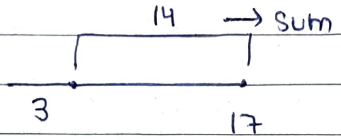
⇉ But how can we find other answers.

⇉ we can do that -

Ex: we are on 17, so, if we subtract 7 from 17 then it's 10. ~~but it is not divisible by 7 then 17~~. Now, we will find 10 in the prefix array. If we find then it's our answer. Otherwise not.

Date _____ Page _____

⇒ After that, we will subtract 14 from 17 i.e we ~~get~~ 3 & 3 is present in array. So, it our one of the answer.



⇒ Now,

$17 - 7 = 10$
$17 - 14 = 3$
$17 - 21 = -4$
$17 - 28 = -11$
⋮ ⋮ ⋮

⇒ But, we can't do this again & again. So, we have to find a relationship b/w 7, 14, 21, ...

⇒ If we find the modulus then it's 0. & if —

1 → 1 (1%7 = 1)
8 → 1 (8%7 = 1)
15 → 1 ⋮
22 → 1 ⋮

⇒ So, we will check those no. whose remainder is same while doing modulus by k.

⇒ So, prefix Sum → 3 8 14 17 26 30 36 45
Sum%k → 3 1 0 3 5 2 1 3

⇒ So, we will use previous day technique.

⇒ we will check the no. ~~came~~ come previous

Date _____ Page _____

ly or not & if yes, then how many times.

⇒ Now, how can we solve if we have -ve no.

2	3	-8	-3	11	4	8	6	9	4

k = 7

⇒ Sum → 2 5 -3 -6 5 9 17 23 32 36

Sum%k → 2 5 -3 -6 5 2 3 2 4 1

⇒ Note: Modulus of -ve no. is different in different programming languages.

Ex: -3 % 7 = -3

-3 % 7 = 4

⇒ How? we can represent any num i.e.

num = A * B + rem

-3 = 7 * 0 + (-3)

-3 = 7 * (-1) + 4

⇒ So, ~~w~~ before doing this operation, first check on the compiler.

⇒ Now, if we check that -

-3, 11, 4, 8, 6, 9

⇒ The sum of above no. is divisible by 7. So, there must be relationship b/w -3 & 4 modulus.

⇒ If we add 7 to -3 then we get 4.

⇒ So, we will convert all the -ve no. by adding 7 to +ve.

Date _____ Page _____

* Subarray Product less. than k:

start → end

| 2 | 5 | 10 | 8 | 100 | 1000 | 5 | 15 |

, k = 999

⇉ In brute force, we find subarray then do product & check.

⇉ T.C. → $O(n^2)$

⇉ Using prefix product is not a good approach here to use.

⇉ So, we will start from starting & take window one by one i.e. —

⇉ First, we will take 2, So, product is less than 999, we will increase the count by 1.

⇉ Next, we will take [2|5], now same here product is less than 999, but here we add 2 to the count because alone 5 can also be a answer.

⇉ So, we will add size of array to count.

⇉ If the product exceeds k then we start decreasing the sub array from starting

Code

Date _____ Page _____

```
while( end <n){
    product *= nums[end];
    while( product >= k){ && start <=end
        product /= nums [start];
        start ++;
    }
    count+= end- start +1;
    end ++;
}
```

ct is

gs,
nt

tart
nting