Day - 193

Sliding Window

* Zero Sum subarray:

6	-1	-3	4	-2	2	4	6	-12	-2

⇒ We have to return total no. of Sub arrays that have sum 0.

-1	-3	4	⇒ 0

-2	2	⇒ 0

⇒ So, in brute force approach, we can easily ~~to~~ solve this problem by selecting every element one by one.

Code

```
for ( i=0; i<n; i++){ → n
        for(j=i; j<n; j++){ → n
              int sum = 0;
              for( k= i ; k<=j ; k++) → n
                    sum += arr[k];
              if (sum==0) total++;
        }
}
```

⇒ $O(n^3)$ → T.C.

⇒ We can reduce the T.C. to $O(n^2)$ by adding directly to the sum, when we arrive to any element.

```
for (i=0; i<n; i++){
        int sum =0;
        for(j=i; j<n; j++){
            sum += arr[i];
            if (sum== 0)
                total ++;
        }
}
```

⇒ Now, our task is to solve this problem in O(n).

⇒ To do this, we will use prefix sum.

6	-1	-3	4	-2	2	4	6	-12	-2
6	5	2	6	4	6	10	16	4	2

⇒ We get any any prefix sum then we will check if we got any that prefix sum. If yes, then add that no. of times prefix sum to the total.
Ex: We get 6 & we also 6 in the array. So, increase total by one.

⇒ If we get zero in the prefix sum then we will add 1 in the total.

⇒ So, we will use map for storing the that sum & their count.

⇒ Also, init the map with (0, 1) so that we don't get any error.

⇒ Also, we will use unordered map.

* Subarray sum equals k:

2	1	7	-4	2	1	3	4	-15	2	-3	6

k = 6

prefixSum → 2 3 10 6 8 9 12 16 1 3 0 6

⇒ We can use the previous approach to solve this problem.

⇒ So, we ask on every prefix —

number - 6 = result

⇒ If this result is in the prefix sum before that no., add that times to the total.

Ex:

2 (prefixSum) - 6 (k) = -4

⇒ Now, we check, if we got -4 before 2 in the prefix sum or not.

⇒ If yes, then add the no. of times of -4 in the total.