100-Days_of_DSA

* for printing all the Subarrays

```
for(int sp=0; sp<N; sp++)   //sp: start point
{
  for(int ep=0; ep<N; ep++)   //ep: end point
  {
    for(int i=sp; i<=ep; i++)
    {
      print(arr[i]);
    }
  }
}
```

* If you want to print Sum of Each Subarray...

```
int psum[] = prefixSum(arr);
```

psum[] ← prefixSum of arr

```
for(int sp=0; sp<N; sp++)
{
  for(int ep=sp; ep<N; ep++)
  {
    if(sp==0)
      print(psum[ep]);
    else
      print(psum[ep]-psum[sp-1]);
  }
}
```

// Run a loop of sp → (0 to n)

ep → (sp to n)

→ // [sp, ep] → Our valid Subarray

*** Contribution Technique ***

arr : { 5 3 -1 8 }

5			
5 3		3	
5 3 -1		3 -1	-1
5 3 -1 8		3 -1 8	-1 8 | 8

Subarrays Starting with (5) (3) (-1) (8)

Value	Occurance
arr[0] = 5	4 times
	⇒ 5 * 4 = 20
arr[1] = 3	6 times
	⇒ 3 * 6 = 18
arr[2] = -1	6 times
	⇒ -1 * 6 = -6
arr[3] = 8	4 times
	⇒ 4 * 8 = 32

ans = 20 + 18 - 6 + 32 = 64

Generalising,

arr[4] : {5 3 -1 8}

no. of times appeared : x_0 x_1 x_2 x_3

ans of 0^{th} idx

$$ans = (arr[0] * x_0) + (arr[1] * x_1)$$

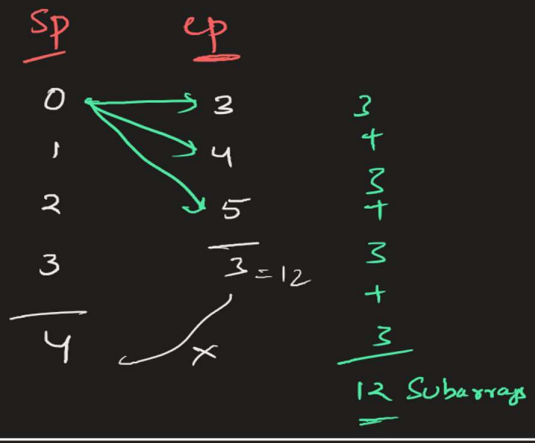
$$+ (arr[2] * x_2) + \dots$$

x_n: no. of times Occured

Finding Occurrance

Ex :- { 3 -2 4 (-1) 2 6 }
0 1 2 3 4 5

A Subarray which includes (-1) is having :



∴ if 'i' is 3

$$\text{Ans} \Rightarrow (i+1) * (n-i)$$

for x_0, $n = 4$

$(i=0) \rightarrow x_0 = (0+1) * (4-0)$

$= 1 * 4 = 4$

Likewise...

```
//pseudo code

int ans = 0;

for(int i=0 ; i<n; i++)

{
    int occ = (i+1)*(n-i);

    ans += (occ * arr[i]);

}
```