Subarrays

- continous part of an array
- single elements / complete array are subarrays
- empty [ ] is not a subarray.

	0	1	2	3	4	5	6
Ex	3	4	5	6	-2	8	10

indices	
2 3 4	✓
0 1 3 4	✗
6	✓
0 , 6	✗
0, 1, 2, 3, 4, 5, 6	✓

# of subarrays.

	0	1	2	3	4	5	6
arr[7] =	4,	2,	10,	3,	12,	-2,	15

Subarrays starting from index 0^{th} = [0,0] [0,1] [0,2] . . . [0 6]
= 7.

Subarrays starting from index 1^{st} = [1, 1] [1, 2] , [1, 3] - - - [1, 6]
= 6

Subarrays starting from index 2^{nd} = [2,2] [2,3] [2,4] . . . [2,6]

= 5.

N size array

Start at 0^{th} = (0,0) (0,1) (0,2) . . . (0, N-1)   N

Start at 1^{st} = (1,1) (1,2) . . . (1, N-1)   N-1

2^{nd} =   N-2

⋮   ⋮

$(N-1)^{th}$ =   1

Total subarrays = 1 + 2 + 3 + . . . N-1 + N

$$= \frac{N(N+1)}{2}$$

Q. Print all value of given subarray.

```
void printsub( int arr[], int s, int e)
{
    for( i=s ; i<=e ; i++)
    {
        print (A[i])
```

TC: O(N)

SC: O(1)

O(e-s+1)

Q. Given subarray, Cal sum of all ele in the subarray.

```
int sumsub( int arr[], int s, int e)
{
        sum = 0
        for( i = s ; i <= e ; i++)
        {
            sum = sum + arr[i]
        }
        return sum.
}
```

TC: O(N)

O(e-s+1)

SC: O(1)

Q. Print all the subarrays.

$$A = \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ [2, 8, 9] \end{array}$$

[0,0] ⇒ 2

[0,1] 2 8

[0-2] 2 8 9

[1,1] 8

[1,2] 8 9

2-2 9

```
for( i=0 ; i<n ; i++ )
{
    // i is starting point

    for( j=i ; j<N ; j++ )
    {
        // i is starting & j is ending.

        printsub( arr, i, j )

    }
}
```

TC : O(N^3)

SC : O(1)

0 0
0 1
0 2
0 3
0 N-1
1 1
1 2
1 3
1 N-1

2 :
|
|
|
|
|

Q. Print sum of every subarray.

$$A = \begin{array}{ccc} 0 & 1 & 2 \\ 3 & -2 & 4 \end{array}$$

0 ~ 0 = 3
0 ~ 1 = 1
0 ~ 2 = 5
1 ~ 1 = -2
1 ~ 2 = 2
2 ~ 2 = 4

```
for( i=0 ; i<n ; i++ )
{
    // i is starting point
    for( j=i ; j<N ; j++ )
    {
        // i is starting & j is ending

        sum = sumsub( arr, i, j )
        print(sum)
    }
}
```

TC: $O(N^3)$
SC: $O(1)$

$$\begin{array}{rccc} & 0 & 1 & 2 \\ A = & 3 & -2 & 4 \\ PF = & 3 & 1 & 5 \end{array}$$

// PF array

```
for( i=0 ; i<n; i++ )
{
    // i is starting point
    for( j=i ; j<N ; j++ )
    {
        // i is starting & j is ending

        sum = pref[j] - pre[i-1]   <-  i > 0

            sum = pref[j]              i == 0
    }
}
```

TC: O(N^2)
SC: O(N)

0 - 0
0 - 1
0 - 2
1 1
1 2
2 - 2

3
1
5
-2
2
4

TC O(N^3)
SC O(1)

| PF

TC : $O(N^2)$
SC : $O(N)$

TC : $O(N^2)$
SC : $O(1)$

Q. Print sum of all subarrays starting from index 2.

0	1	2	3	4	5	6
7	3	2	-1	6	8	2

[2 2] = 2
[2, 3] = 1
[2, 4] = 7
[2, 5] = 15
[2, 6] = 17

```
sum = 0

for ( i = 2; i < n; i++ )
{
    sum = sum + arr[i]
    print (sum)
}
```

TC : $O(N)$
SC : $O(1)$

Q. Print sum of all subarrays starting from index 3.

```
sum = 0
for ( i = 3 ; i < n ; i++ )
{
    sum = sum + arr[i]
    print (sum)
```

TC : O(N)
SC : O(1)

Q. Print sum of all subarrays starting from index i.

```
sum = 0
for ( j = i ; j < n ; j++ )
{
    sum = sum + arr[j]
    print (sum)
```

TC : O(N)
SC : O(1)

Q print all subarray sum

```
for( i=0; i<n; i++)
{
    // print all subarrays starting from i

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

TC: $O(N^2)$
SC: $O(1)$

TC $O(N^3)$ SC: $O(1)$

$\downarrow$ PF sum

TC: $O(N^2)$ SC: $O(N)$

$\downarrow$ carry forward

TC: $O(N^2)$ SC: $O(1)$

Q. Given an array, find sum of all subarray sums.

$$\begin{array}{cccc} & 0 & 1 & 2 \\ A = & 3 & -1 & 4 \end{array}$$

[0,0] : 3	3		
[0,1] = 2	3	-1	
[0,2] = 6	3	-1	4
[1,1] = -1	-1		
[1,2] = 3	-1	4	
[2,2] = 4	4		

$3 \times 3 + 4(-1) + 3(4) = 17$

$3 + 2 + 6 - 1 + 3 + 4 = 17$

```
ans = 0
for( i = 0 ; i < n ; i++ )
{
    // print all subarrays starting from i

    sum = 0
    for ( j = i ; j < n ; j++ )
    {
        sum = sum + arr[j]
        ans = ans + sum
    }
}
return ans.
```

TC : $O(N^2)$
SC : $O(1)$

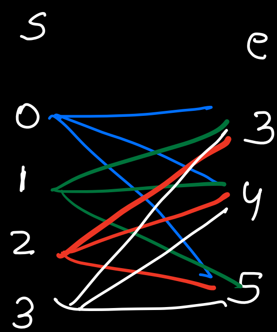
Q. In how many subarrays index 3 present

	0	1	2	3	4	5
arr[] =	3	-2	4	-1	2	6

$N = 6$

$(i+1)(N-i)$

4×3



[0,3] [0,4] [0,5]

(1, 3) (1, 4) (1, 5)

(2, 3) (2, 4) (2, 5)

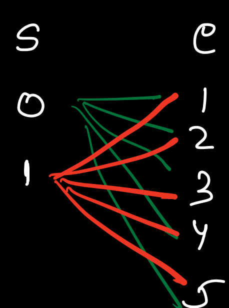
(3, 3) (3, 4) (3, 5)

Total subarray

$4 \times 3 = 12$

Q. In how many subarrays index 1st present

	0	1	2	3	4	5
arr[] =	3	-2	4	-1	2	6



$2 \times 5 = 10$

Q. In how many subarrays index i^{th} present

Array size = N.

index 0 1 2 3 . . . . $i-1$ i $i+1$ . . . $N-1$

s	e
0	i
1	$i+1$
2	$i+2$
3	⋮
⋮	⋮
i	$N-1$

$$\# = \begin{pmatrix}\text{count} \\ \text{of} \\ s\end{pmatrix} * \begin{pmatrix}\text{count} \\ \text{of} \\ e\end{pmatrix}$$

$$= (i+1) * (N-i)$$

$$N - \cancel{1} - i + \cancel{1}$$

$$N - i$$

Q. Given an array, find sum of all subarray sums.

$$\begin{array}{cccc} & 0 & 1 & 2 \\ A = & 3 & -1 & 4 \end{array}$$

$[0,0]$: 3	3	
$[0,1]$ = 2	3	−1
$[0,2]$ = 6	3	−1 4
$[1,1]$ = −1	−1	
$[1,2]$ = 3	−1	4
$[2,2]$ = 4	4	

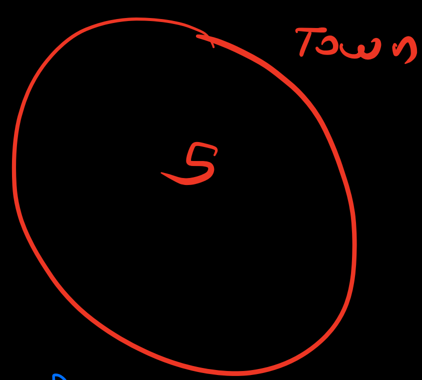
$3 \times 3 + 4(-1) + 3(4) = 17$

$3 + 2 + 6 - 1 + 3 + 4 = 17$

Mela (fair)

Mon	3
Tue	4
Wed	1
Thu	2
Fri	5

15



$P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5$

$2 + 3 + 0 + 0 + 0 = 15$

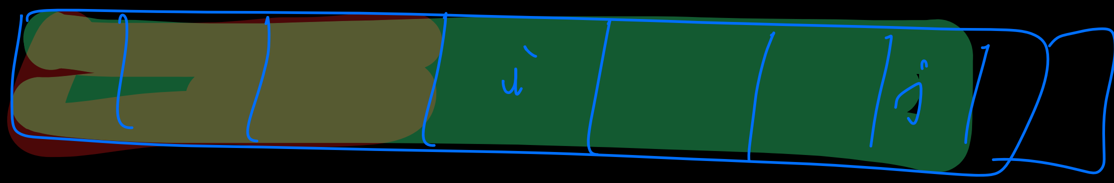
Total ans = Contribution of 0^{th} index + Contri of 1^{st} inde + - - - ——

= (No of times 0^{th} index) * arr[0] + (No of times i^{th} index) * arr[i] + - - -

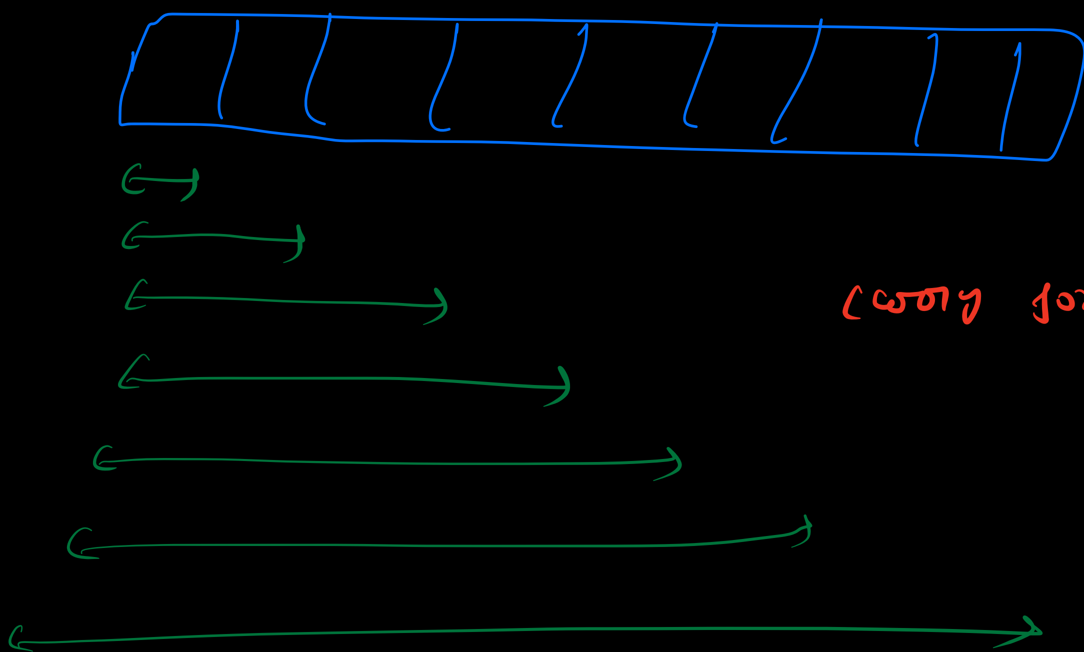
```
int ans = 0
for( i = 0; i < n; i++)
{
    // add contri of i to ans.
    no of times = (i+1)(n-i)
    contri of i = (no of times) * arr[i]
    ans = ans + contri of i
}
return ans.
```

TC: O(N)
SC: O(1)

Contribution Technique



$sum(i,j) = pref[j] - pref(i-1)$



carry forward

ans = contrib of 0th + contrib of 1st + ...