Welcome ☺

Agenda : Contribution Technique
Sliding Window

Q Find the sum of all subarray sums

eg: I/p ⇒ [3, 2, 5]

3	⇒	3
3, 2	⇒	5
3, 2, 5	⇒	10
2	⇒	2
2, 5	⇒	7
5	⇒	5
		32

Brute force

Use 3 nested for loops. Two loops will generate all the possible subarrays and third will be used to calculate sum of the subarray

T.C ⇒ $O(N^3)$ S.C ⇒ $O(1)$

App 1 Prefix Sum technique.

1. Calculate prefix sum
2. Two loops to get all subarrays
3. Get subarray sum using prefix sum array.
4. Add subarray sum to total sum

T.C ⇒ $O(N^2)$
S.C ⇒ $O(N)$

App2 Carry Forward Technique.

A: [-4, 1, 3, 2]

-3

i=0
j ⇒ 0 to 3

i	j	Sum
0	0	-4
0	1	-4+1 = -3
0	2	-3+3 = 0
0	3	0+2 = 2

Code

```
totalSum = 0
for( i=0 ; i< N ; i++)
{
    sum = 0
    for( j=i ; j<N ; j++)
    {
        sum = sum + A[j]
        totalSum = totalSum + sum
    }
}
```

A: [-4, 1, 3, 2]

i	j	sum	totalSum
0	0	0 -4	-4
0	1	-4+1=-3	-4-3 = -7
0	2	-3+3=0	-7+0 = -7
0	3	0+2=2	-7+2 = -5
1	1	0+1=1	-5+1 = -4

⇒ We are able to use carry forward technique without using extra space b/c queries are sequential.

App3 Contribution Technique

A: [1 2 3]

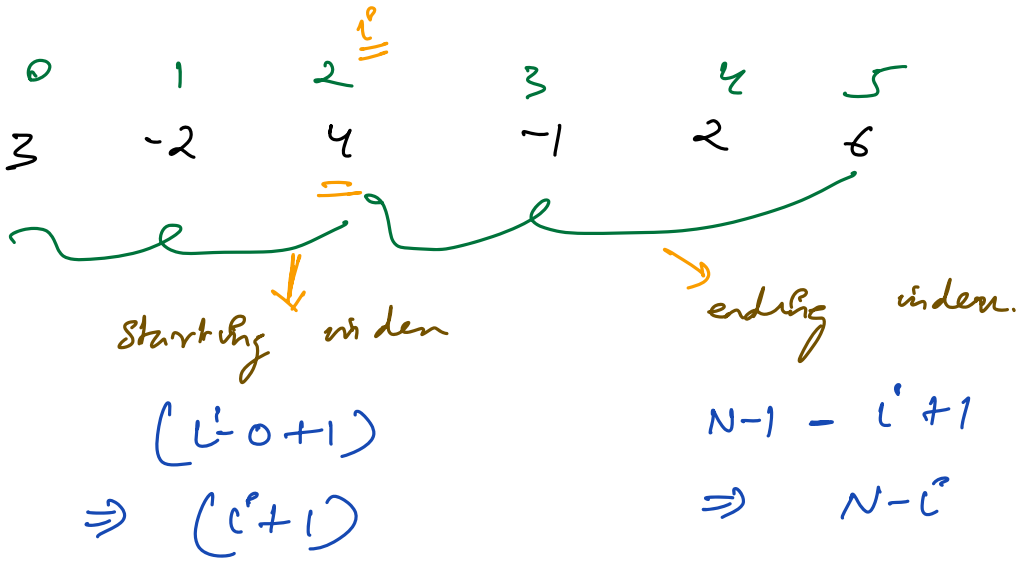
1	→	1
1 , 2	→	3
1 , 2 , 3	→	6
2	→	2
2 , 3	→	5
3	→	3
		20

Contribu" of "1" in totalSum = 1 * 3 = 3

Contribu" of "2" in totalSum = 2 * 4 = 8

Contribu" of "3" in totalSum = 3 * 3 = 9

3 + 8 + 9 = 20



Contribu" of i^th element ⇒ (N-i)(i+1)

Final contribu" ⇒ (N-i)(i+1) * A[i]

Code

```
for ( i -> 0 to N-1)
{
    totalSum += arr[i] * (i+1)(N-i)
}
return totalSum.
```

T.C → O(N)
S.C → O(1)

A: [1 2 3]

i°	contribⁿ of ith	totalSum.
0	1 * 3 = 3	3
1	2 * 4 = 8	3 + 8 = 11
2	3 * 3 = 9	11 + 9 = 20

Count of subarrays of length K

		N-K			N-1
0	1	2	3	4	5
3	-2	4	-1	2	6

K=4 ✓ ✓ ✓ ✗ ✗ ✗

K=5 ✓ ✓ ✗ ✗ ✗ ✗

range of starting index [0, N-K]

# subarrays of len K	=	N-K+1

Q Given an array of N elements. Print max. subarray sum for subarrays with length K.

eg:

0	1	2	3	4	5	6	7	8	9
-3	4	-2	5	3	-2	8	2	-1	4

K = 5

SI	eI	sum
0	4	7
1	5	8
2	6	12
3	7	16 ⇐
4	8	10
5	9	11

Bruteforce Calculate sum of all subarrays of size K and find maximum out of them.

T.C → O(N-K+1) * K ⇒ O(N²)

S.C → O(1)

App 1 Sliding Window Technique. (similar to carry forward with slight change)

⇒ 1: Add new element to curr sum.

2: Subtract the first element of curr. subarray.

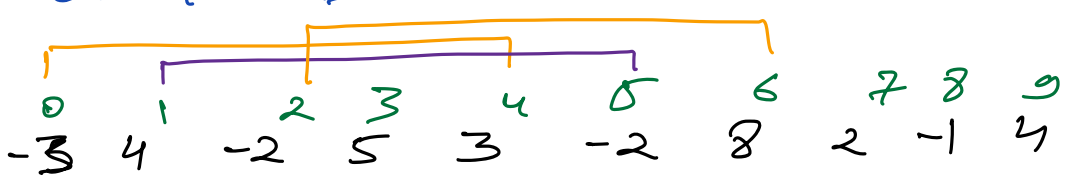
⇒ Use sliding window for Fixed length subarray

Code

```
func ()
{
    i = 0
    j = K-1

    // 1st window.
    for ( m → 0 to K-1)
    {
        sum += A[m]
    }
    ans = sum
    i++
    j++          } shifted the window.

    while ( j < N)
    {
        sum = sum + A[j] - A[i-1]
        ans = max ( sum, ans)
        i++
        j++
    }

    return ans
}
```

add new element

remove first element



K = 5

i	j	sum	ans
0	4	7	7
1	5	7 + (-2) - (-3) = 8	8
2	6	8 + 8 - 4	12

TC ⇒ O(N)

SC ⇒ O(1)