Welcome ☺

Agenda : Subarrays
2-3 problems

Subarrays → contiguous part of the array in order

	0	1	2	3	4	5	6	7	8
arr :	4	1	2	3	-1	6	9	8	12

eg : 3 2 1 4 ✗ * reverse order not considered subarray
eg : 2 3 -1 6 ✓

4 12 ✗

1 2 6 ✗

9 ✓

single element ✓

whole array ✓

Represent a subarray

start index
end index
→ uniquely represent a subarray

start ≤ end

Q How many subarrays are starting from index 0.

SI

4 2 10 3 12 -2 15

Ans = 7

Q How many subarrays are starting from index 1.

4 2 10 3 12 -2 15

Ans = 6

Total subarrays for all index

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

Q Print subarray

```
printsubarray ( startIndex , endIndex)
{
    for ( i = startIndex ; i <= endIndex ; i++)
    {
        print ( arr[i])
    }
}
```

Sum Subarray

```
printsubarray ( startIndex , endIndex)
{
    sum = 0
    for ( i = startIndex ; i <= endIndex ; i++)
    {
        sum += arr[i];
    }
    return sum
}
```

Question 1 Given an array of size N, print all possible subarrays.

eg: 2 8 9

SI	EI	
0	0	(0,0) (0,1) (0,2)
0	1	(1,1) (1,2)
0	2	(2,2)
1	1	
1	2	
2	2	

0,0 → 2

0,1 → 2, 8

Pseudo code

```
// first set the start index
for( int i=0 ; i<N ; i++) -> start Index
{
    for( j=i ; j<N ; j++) -> end Index
    {
        for( k=i ; k<=j ; k++ -> loop over all elements
        {
            print(arr[k]);
        }
    }
}
```

T.C → $O(N^3)$

S.C → $O(1)$

Question 2 Find sum of each and every subarray.

eg: $\overset{0}{3}$ $\overset{1}{2}$ $\overset{2}{-1}$ $\overset{3}{5}$

	sum
0,0	3
0,1	3+2 = 5
0,2	4
0,3	9
1,1	2
⋮	⋮

⇒ Not possible to do it in $O(N)$ time.
b/c # subarrays → $O(N^2)$

Brute force

→ $O(N^3)$

Optimised sol^n

```
// Use Prefin Sum
// Create prefin sum array.
for( int i=0 ; i<N ; i++)
{
    for( int j= i ; j<N ; j++)
    {
        if( i==0)  sum = pf[j] ;
        else sum = pf[j] - pf[i-1]

    }
}
```

T.C → $O(N^2)$
S.C → $O(N)$

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

Sum of subarrays starting from index 2

sum = 0

2,2 arr[2] = 2

2,3 arr[2] + arr[3] = 1

2,4 arr[2] + arr[3] + arr[4]

2,5

// carry forward

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

T.C → O(N²)

S.C → O(1)

// we are able to use prefix sum idea directly without using extra space. b/c queries are sequential

4 2 1 3

i	j	sum
0	0	0+4
0	1	4+2=6
0	2	6+1=7
0	3	7+3=10
1	1	0+2
1	2	2+1=3
1	3	3+3=6
2	2	0+1=1
2	3	1+3=4
3	3	0+3=3

Question Find total sum of all subarray sums.

Google Meta

eg: 3 2 -1

i	j	sum
0	0	3
0	1	5
0	2	4
1	1	2
1	2	1
2	2	-1
		14

T.C → O(N²)
S.C → O(1)

```
totalSum = 0 ;
for( int i = 0;  i<N ; i++)        SI
{   sum = 0
    for ( int j=i ;  j<N ; j++)    EI
    {
        sum += arr[j];
        totalSum += sum;
    }
}
```

Can we optimize it further ?

eg: -1 3 4 (indices 0 1 2)

0,0	arr[0]
0,1	arr[0] + arr[1]
0,2	arr[0] + arr[1] + arr[2]
1,1	arr[1]
1,2	arr[1] + arr[2]
2,2	arr[2]

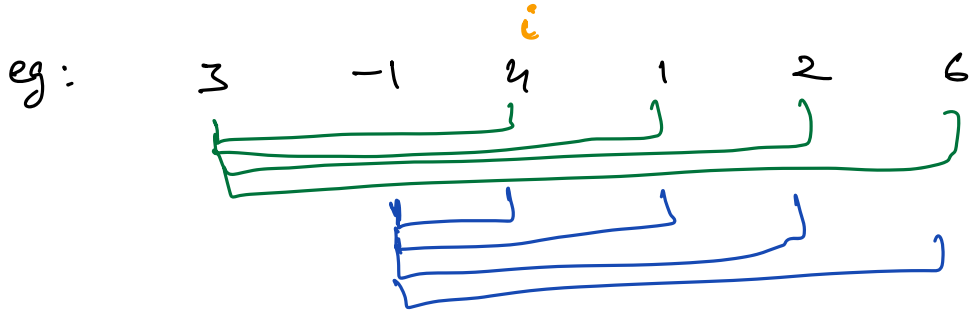
$3 * arr[0] + 4 * arr[1] + 3 * arr[2]$

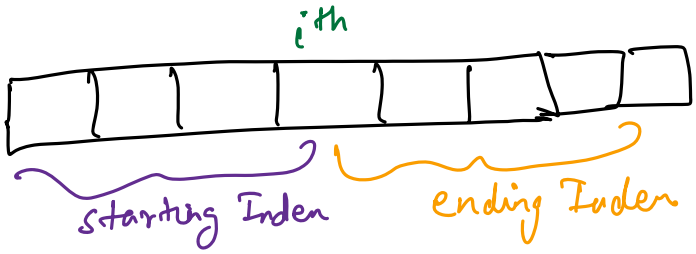
$3*(-1) + 4*(3) + 3*(4)$

21

# of particular element in all subarrays

// In how many subarrays a particular element 'i' is present.





SI $\Rightarrow$ $0 \rightarrow i$

EI $\Rightarrow$ $i \rightarrow n-1$

contribn of i^{th} element $= (i+1) * (n-i)$

$\downarrow\downarrow$

Final contribn $= (i+1) * (n-i) * arr[i]$

H.W