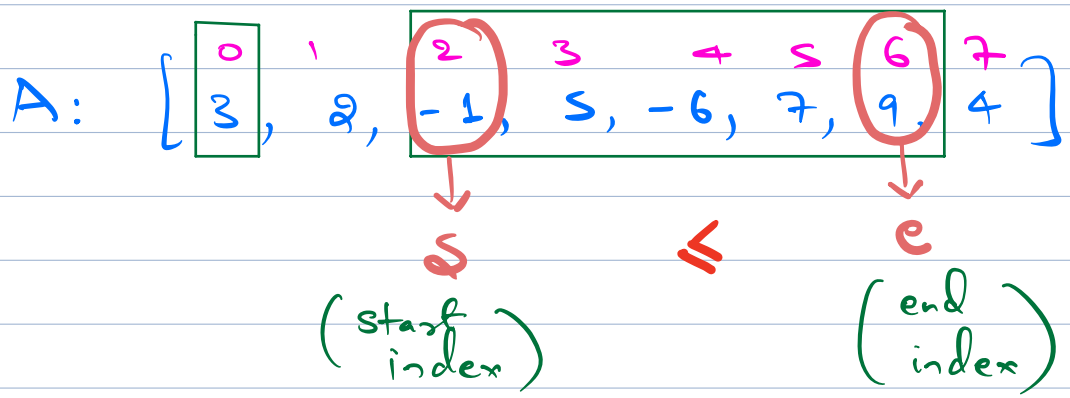
Subarrays

⇒ Contiguous part of an array.



[4, 5, 1, 9, 0, 2, 3, 5]

a (5) ✓

~~b~~ [4, 5, 1, 0]

c [9, 0, 2, 3] ✓

d [4, 5, 1] ✓

Largest Subarray → s = 0, e = N-1 > Entire array.

Smallest subarray ∀ single elements (s = e)

	0	1	2	3	4	5	6
A:	4	2	10	3	12	-2	15

Q Is [3, 12, 15, -2] a subarray.

No ⇒ order is different.

Q Given an array of size N. find the no. of subarrays possible ??

	0	1	2	3	4	5	6
A:	4	2	10	3	12	-2	15

$s \le e$

s	e		Count
0	[0 - 6]	=	7
1	[1 - 6]	=	6
2	[2 - 6]	=	5
3	[3 - 6]	=	4
4	[4 - 6]	=	3
5	[5 - 6]	=	2
6	[6 - 6]	=	1

$7 + 6 + 5 + 4 + 3 + 2 + 1 = 28$

Sum of 1st N natural no.

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

$$= O(N^2)$$

Print all the values of a given subarray from l to r.

Code

```
void printSubarray ( A[], l, r) {
    for(i = l; i <= r; i++) {
        print (A[i]);
    }
}
```

Print all the subarrays of a given array.

A: [2, 8, 9] (indices 0, 1, 2)

s	e		
0	0	⇒	[2]
0	1	⇒	[2, 8]
0	2	⇒	[2, 8, 9]
1	1	⇒	[8]
1	2	⇒	[8, 9]
2	2	⇒	[9]

Code

```
for (s = 0; s < N; s++) {
    for (e = s; e < N; e++) {
        // Subarray [s, e]
        printSubarray {
            for (i = s; i <= e; i++) {
                print (A[i]);
            }
        }
    }
}
```

T.C. = $O(N^3)$

$= (N^2) \times N$

↓ # subarrays ↓ Print

Print the sum of every single subarray.

A: [2, 8, 9] (indices 0, 1, 2)

s	e		Subarray	Sum
0	0	⇒	[2]	(2)
0	1	⇒	[2, 8]	(10)
0	2	⇒	[2, 8, 9]	(19)
1	1	⇒	[8]	(8)
1	2	⇒	[8, 9]	(17)
2	2	⇒	[9]	(9)

1) Brute force

```
for (s = 0; s < N; s++) {
    for (e = s; e < N; e++) {
        // Subarray [s, e]
        sum = 0;
```

$O(N^2)$

```
for (i = s; i <= e; i++) {
    sum += A[i];
}
}
```

T.C. = $O(N^3)$

S.C. = $O(1)$

2) Optimise ??

Definitely req. to define every subarray

```
for (s = 0; s < N; s++) {
    for (e = s; e < N; e++) {
        // Subarray [s, e]
```

$O(N^2)$

Prefix Sum Array

$O(1)$

```
        if (s == 0) {
            print (P[e]);
        }
        else {
            print (P[e] - P[s-1]);
        }
    }
}
```

T.C. = $O(N^2)$

S.C. = $O(N)$ ⇒ Prefix Sum Array.

(Is it possible to reduce) S.C. to $O(1)$?!

(w/o modifying I/P)

Q. Print the sum of every subarray starting on the index 3.

0	1	2	3	4	5	6	7	8	9
15	2	9	6	-2	5	0	1	4	-3

A:

S = 3

E = 3 ⇒ 6 = 6

4 ⇒ 6 + (-2) = 4

5 ⇒ 6 + (-2) + 5 = 9

6 ⇒ 9 + 0 = 9 [Sum[3-5] + A[6]]

7

8

9

Code

```
S = 3;

Sum = 0;
for (e = s; e < N; e++) {

        Sum = Sum + A[e];
        print (sum);

}
```

T.C. = O(N)
S.C. = O(1)

Q Can you use the above sol^n to print sum of every subarray optimally.

Code

```
for (s = 0; s < N; s++) {

        Sum = 0;
        for (e = s; e < N; e++) {

                Sum = Sum + A[e];
```

print (sum);

} }

A: [7, 9, 8] (0, 1, 2)

s = 0 (sum = 0 + 7 = 7 + 9 = 16 + 8 = 24

c = 0 ⇒ 7
c = 1 ⇒ 16
c = 2 ⇒ 24

s = 1 (sum = 0 + 9 = 9 + 8 = 17

c = 1 ⇒ 9
c = 2 ⇒ 17

T.C. = $O(N^2)$
S.C. = $O(1)$

Q Given an array of size N find the sum of all subarray sums.

Google
FB
Amazon
Walmart

{ 1, 2, 3 [0 - 2]

A: [1, 2, 3] (0, 1, 2) Ans = 20

s	e	Sum	
0	0	1	⇒ A[0]
0	1	3	⇒ A[0] + A[1]
0	2	6	⇒ A[0] + A[1] + A[2]
1	1	2	⇒ A[1]
1	2	5	⇒ A[1] + A[2]
2	2	3	⇒ A[2]

E = 20

3 × A[0]
+ 4 × A[1]
+ 3 × A[2]

Soln ⇒ Using Carry forward

```
totalSum = 0;
for (s = 0; s < N; s++) {

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

A:

0	1	2	3	4
1,	2,	9,	6,	-2

total Sum = 0

s = 0	e = 0	Ts = 1
	1	Ts = 3
	2	Ts = 12
	3	Ts = 18
	4	Ts = 16
		49

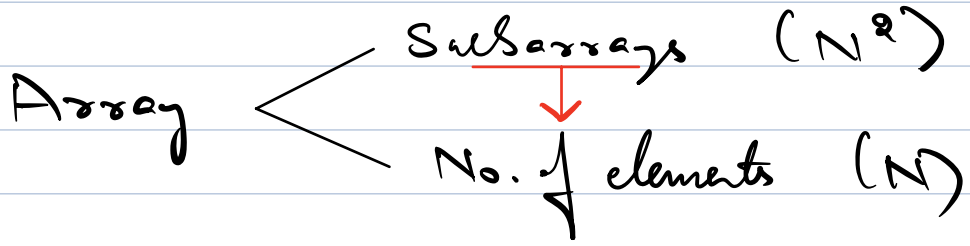
$T.C. = O(N^2)$

$S.C. = O(1)$

Is it possible to optimise the T.C. to $O(N)$??

⇒ If to solve the above problem, we are calculating sum for every subarray we cannot optimise T.C. better than $O(N^2)$

because # subarrays = $O(N^2)$

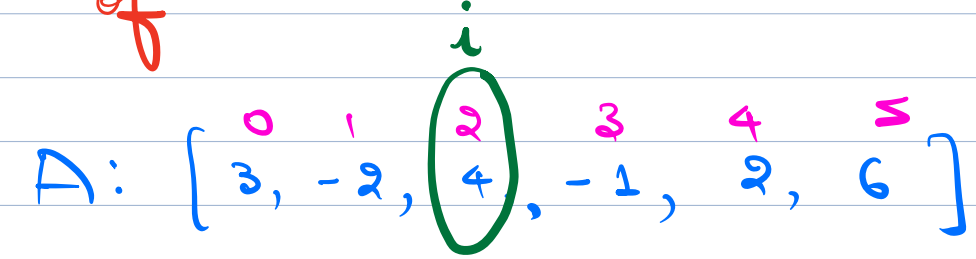


Contribution Technique

⇒ To calculate the contribution of every element to the result.

Contribution towards Total Sum = Total no. of subarrays an element is a part of × Value of the element.

Total no. of subarrays an element on the index i is a part of



$s = [0, 1, 2]$

0,

$s <= i \Rightarrow [0, i] \Rightarrow (i+1)$

$e = [2, 3, 4, 5]$

$e >= i \Rightarrow [i, N-1] \Rightarrow (N-i)$

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

& → multiply
OR → Add

$$\#\text{Subarrays} = \left(\#\text{poss. of } s\right) \& \left(\#\text{poss. of } e\right)$$

$$\left(\#\text{poss. of } s\right) \times \left(\#\text{poss. of } e\right)$$

$$\#\text{Subarrays an element on index } i \text{ is part of} = (i+1) \times (N-i)$$

Code

```
totalSum = 0;

for (i = 0; i < N; i++) {

    count = (i+1) * (N-i);
    contri = count * A[i];

    totalSum += contri;
}
```

T.C. = O(N)
S.C. = O(1)

Doubt

1) > 5 mins to read & understand (Dry Run)
2) 10 mins (max) for Brute force
3) 30 mins for optimised. (max)
4) Look at the solution approach & give another 20 mins.
5) Take Help

$A = [1, 4, 5, 3, 2, 6]$ $N = 6$

$B = 3$

$[0, B-1]$

1) $[0, 1, 2]$ $[\,]$ 10
2) $[0, 1]$ $[5]$ $10 - 5 + 6 = 11$
3) $[0]$ $[5, 4]$ $11 - 4 + 2 = 9$
4) $[\,]$ $[5, 4, 3]$ $9 - 1 + 3 = 11$

Even Subarray.

length = Odd ⇒ ✗