Subarray Basics

→ Continuous part of an array.

1) Single Element ✓

2) Full array. ✓

3) Empty array ✗

Ex1 arr[] =

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

indices	Subarray.	
[1, 2, 3, 4]	✓	= [1, 4]
[1, 2, 4]	✗	=
[4, 5]	✓	= [4, 5]

length of a subarray [i, j] = $j - i + 1$

No of Subarrays in a arr of length n.

$$arr[] = \underset{0}{[} \underset{0}{a_1} \quad \underset{1}{a_2} \quad \underset{2}{a_3} \quad \underset{3}{a_4} \quad \underset{4}{a_5}]$$

Start	no of ending points	
0	5	[0, 1, 2, 3, 4]
1	4	[1, 2, 3, 4]
2	3	[2, 3, 4]
3	2	[3, 4]
4	1	[4]

Start	ending points	count
0	[0, n-1]	n
1	[1, n-1]	n-1
2	[2, n-1]	n-2
⋮		
n-1	[n-1, n-1]	1

$$\frac{n(n+1)}{2}$$

[s, e] Q Print the subarray.

```
for (int i = s ; i <= e ; i++) {
    print (arr[i]);
}
```

⇒ Print all subarrays

```
for (int s = 0 ; s < n ; s++) {          // s <= n-1
    for (int e = s ; e < n ; e++) {      // e <= n-1
        for (int i = s ; i <= e ; i++) {
            print (i);
        }
    }
}
```

TC: $O(N^3)$

SC: $O(1)$

Q Print all Subarray Sum.

```
Sum = 0;
for (int s = 0 ; s < n ; s++) {
    for (int e = s ; e < n ; e++) {
        Sum = 0;
        for (int i = s ; i <= e ; i++) {
            Sum += arr[i];
        }
        Print Sum;
    }
}
```

TC: $O(n^3)$

SC: $O(1)$

Optimization using prefix Sum.

(Pf[])

// Build the Prefix Sum Array. ⇒ O(n)

```
for (int s = 0 ; s < n ; s++) {
    for (int e = s ; e < n ; e++) {
        if (s == 0)
            print pf[e];
        else
            print pf[e] - pf[s-1];
    }
}
```

O(n²)

TC: O(n) + O(n²) ≃ O(n²)

SC: O(n) ⇒ if you do not overwrite the original array.

SC: O(1) ⇒ if original array is modified.

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

Print all subarray sum for subarrays starting at index = 2

⇒ [5] = 5

[5 6] = 11

[5 6 -1] = 10

```
Sum = 0
for (int i = 2 ; i < n ; i++) {
    Sum += arr[i];
    print sum;
}
```

TC : O(n)
SC : O(1)

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

TC : $O(n^2)$

SC : $O(1)$

Print max subarray Sum

[n = 100]

A[i] = -10

```
max_Sum = INTEGER.MIN.
for (int s = 0 ; s < n ; s++) {
    sum = 0
    for (int i = s ; i < n ; i++) {
        sum += arr[i];
        max_Sum = max(max_Sum, sum);
    }
}
```

TC : $O(n^2)$

SC : $O(1)$

KADANE

It can be optimised to $O(N)$

Q Print all subarray sum.

Ex arr[] = [1, 2]

[1] = 1
[2] = 2
[12] = 3
6

```
total_sum => 0
for (int s = 0 ; s < n ; s++) {
    sum = 0
    for (int i = s ; i < n ; i++) {
        sum += arr[i];
        total += sum;
    }
}
return total_sum;
```

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

Q

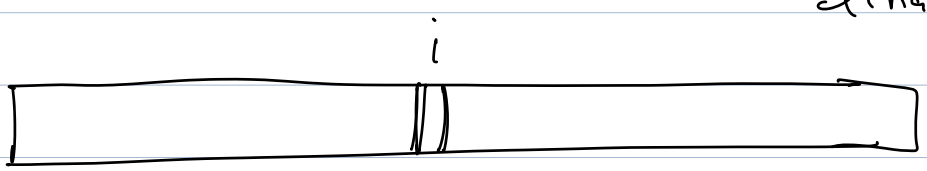
arr[] = [1 2 4 6 1 5] (indices 0 1 2 3 4 5)

start ⇒ {0, 1, 2} = 3

end ⇒ {2, 3, 4, 5} = 4

Total ⇒ start × end ⇒ 12

length ⇒ n



start ⇒ [0, i] ⇒ i − 0 + 1 = i + 1

end ⇒ [i, n−1] = n − 1 − i + 1 = n − i

Total ⇒ (i + 1) × (n − i)

$$arr[] = \underset{0}{[} \underset{0}{2} \quad \underset{1}{4} \quad \underset{2}{1} \quad \underset{3}{2} \quad \underset{4}{5}]$$

$s \Rightarrow 1 \Rightarrow \boxed{5}$

$e \Rightarrow 5$

$$(i+1) \times (n-i)$$

$\Rightarrow 1 \times 5 = 5$

$$arr \quad [\quad -1 \quad 3 \quad 4]$$

$$X \quad Y \quad Z$$

Total Subarray Sum $\Rightarrow (-1)(x) + 3(y) + 4(z)$

$[-1]$	$[3]$	$(-1) + (-1) + (-1)$
$[-1, 3]$	$[3, 4]$	$(3) + (3) + (3) + (3)$
$[-1, 3, 4]$	$[4]$	$4 + 4 + 4$

$1 + (3) + 3(4) + 4(3)$

total_sum = 0

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

int contribution ⇒ (i+1) * (n-i);

total sum += (contribution) x arr[i];

}

return total_sum;

TC: O(n)

SC: O(1)