- Subarray Basics
- Printing subarrays
- Generating subarrays
- Printing all subarray sums
- Max subarray sum
- Sum of all subarray sums
- Maxm subarray sum of length-k.

Basics :-

: continuous part of an array is a subarray

: empty arr is not subarray

: single element is a subarray

ex ⇒ arr[] ⇒ -2 4 6 3 8 1 4 3 2 -10

0 1 2 3 4 5 6 7 8 9

idx ⇒ [3 4 5 7 8] → XX [6 is missing]

idx ⇒ [4 5 6 7 8] → ✓

idx ⇒ [4] → ✓

[] → XX

Q1. Given a startIdx, endIdx print the subarray

ex ⇒ 2 4 6 3 8 9 -1

0 1 2 3 4 5 6

S = 2

e = 5

O/p ⇒ 6 3 8 9

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

$\boxed{\begin{array}{l} TC \Rightarrow O(N) \\ SC \Rightarrow O(1) \end{array}}$

Q2. Given an array, find the no. of subarrays

ex => arr[4] => -1 3 2 3

0 1 2 3

Starting from idx 0 :-
0 - 0 → [-1]
0 - 1 → [-1 3]
0 - 2 → [-1 3 2]
0 - 3 → [-1 3 2 3]
] ④

Starting from idx 1 :-
1 - 1 → [3]
1 - 2 → [3 2]
1 - 3 → [3 2 3]
] ③

Starting from idx 2 :-
2 - 2 → [2]
2 - 3 → [2 3]
] ②

Starting from idx 3 :- 3 - 3 → [3]] ①

total no. of subarrays = 4+3+2+1 → $\frac{(N)(N+1)}{2}$

= 10

Generalise

arr => { 0 1 2 3 - — i — — N-2 N-1 }
N

//start - 0

$$\left.\begin{array}{l} 0-0 \\ 0-1 \\ 0-2 \\ 0-3 \\ \vdots \\ 0-N-1 \end{array}\right] N \text{ subarrays}$$

//start 1

$$\left.\begin{array}{l} 1-1 \\ 1-2 \\ 1-3 \\ 1-4 \\ \vdots \\ 1-N-1 \end{array}\right] N-1 \text{ subarrays}$$

//start - 2

$$\left.\begin{array}{l} 2-2 \\ 2-3 \\ 2-4 \\ 2-5 \\ \vdots \\ 2-N-1 \end{array}\right] N-2 \text{ subarrays}$$

N-1

$(N-1)-(N-1)$] 1 subarrays.

$N + N-1 + N-2 + \ldots + 3 + 2 + 1$

$\Rightarrow 1 + 2 + 3 + \ldots + N-2 + N-1 + N$

$\Rightarrow \boxed{\dfrac{N(N+1)}{2}}$ $\Rightarrow$ o/p => print$\left(\dfrac{N(N+1)}{2}\right)$

TC => O(1)
SC => O(1)

Q 3. Given an array print all subarrays:

arr[3] => [8 2 9]

0 1 2

→ start from 0:

0-0 → [8]

0-1 → [8 2]

0-2 → [8 2 9]

→ start from 1

1-1 → [2]

1-2 → [2 9]

→ start from 2

2-2 → [9].

[s e] $\boxed{s \le e}$

```
for(s=0; s<N; s++){
    for(e=s; e<N; e++){
        // [s-e] → subarray
        for( i=s; i<=e; i++){
            print(arr[i])
        }
    }
}
```

TC => $O(N^3)$

SC => $O(1)$

=> arr => [8 2 9]

0 1 2

s => 0	s => 0	s => 0	s => 1	s => 1	s => 2
e => 0	e => 1	e => 2	e => 1	e => 2	e => 2

0-0	0-1	0-2	1-1	1-2	2-2
8	8 2	8 2 9	2	2 9	9.
✓	✓	✓	✓	✓	✓

Q4, Given an array, print all subarray sums.

arr[3] ⇒ [8 2 9]

0-0 → [8] → 8

0-1 → [8 2] → 10

0-2 → [8 2 9] → 19

1-1 → [2] → 2

1-2 → [2 9] → 11

2-2 → [9] → 9

Brute force

```
for(s=0; s<N; s++) {
        for( e=s; e<N; e++) {
            // [s-e] → subarray
            sum=0;
            for( i=s; i<=e; i++) {
                    sum = sum + arr[i];
            }
          print(sum)
        }
}
```

$$\boxed{\begin{array}{l} TC \Rightarrow O(N^3) \\ SC \Rightarrow O(1) \end{array}}$$

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

sum array [s -e]

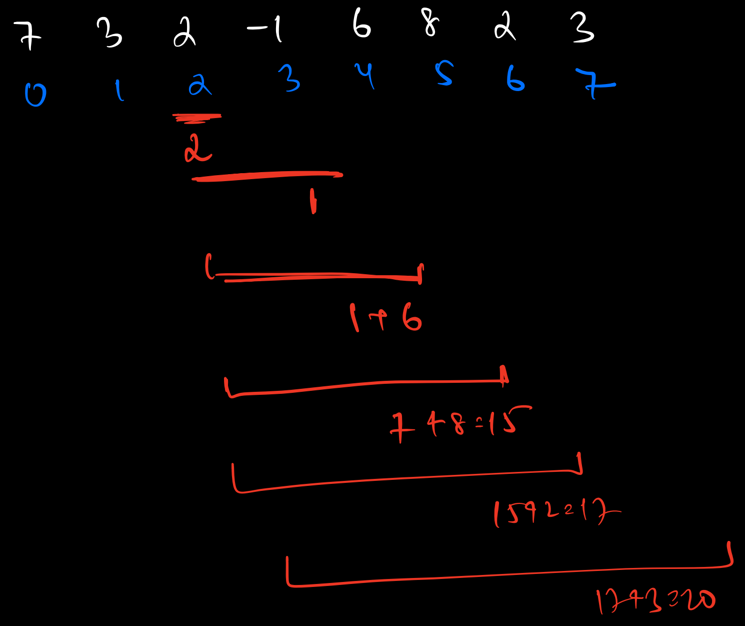
TC ≃ O(N+N2) ≈ O(N2)

SC => O(N).

Q 5. Print all subarray sums, start at idx 2

arr[8] =>	7	3	2	-1	6	8	2	3
	0	1	2	3	4	5	6	7

[2 2]	2
[2 3]	1
[2 4]	7
[2 5]	15
[2 6]	17
[2 7]	20.



Carry forward the sum :-

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

TC => O(N)
SC => O(1)

Q 6. → Print all subarray sums using carry forward.
(optimised Q4)

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

TC => $O(N^2)$
SC => O(1)

Q.7. Find the maxm sum of all subarray sums:-

```
maxSum = INT_MIN
for(s=0; s<N; s++){
    sum = 0
    for(i=s; i<N; i++){
            sum = sum + arr[i]
            maxSum = max(maxSum, sum)
    }
}
 print(maxSum).
```

TC => $O(N^2)$

SC => $O(1)$

this can be solved in → O(N) → Kadane's Algo
↓
discussed in
[advanced lectures]

Q.8 Print sum of all subarray sums:-

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

TC => O(N2)

SC => O(1)

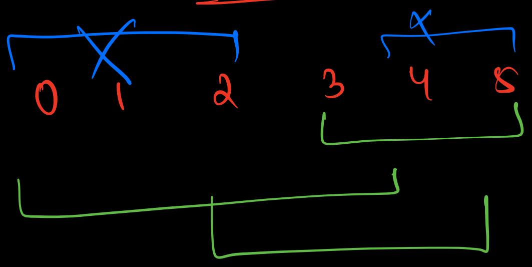
Q9 How many subarrays is idx 3 present ?

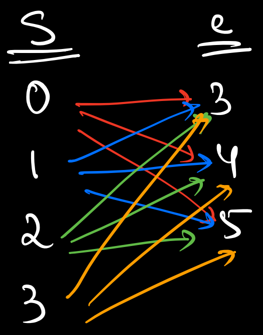
arr ⇒ 3 -2 4 -1 2 6
0 1 2 3 4 5

subarrays ⇒ [0-3] [2 3] [1 3] [~~4 5~~]
[0-4] [3 3] [2 4] [2 5]
[0-5] [~~1-2~~] [~~2/2~~]
[~~0-2~~] X

subarray → [s-e]

$s \le 3$ && $e \ge 3$.





⇒ 4 (no. of s idx)

* 3(no. of e idx)

12

Q 10. How many subarrays contain idx i. ?

$$arr \Rightarrow \begin{array}{cccccc} [3 & -2 & 4 & -1 & 2 & 6]. \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array}$$

$i = 1.$

$s \leq 1$	$e \geq 1.$
0	1
1	2
	3
	4
	5

$\Rightarrow 10 \ (5*2)$

$[a \ \ b)$

$= \underline{\underline{b-a+1}}$

$$[\underbrace{0 \quad 1 \quad 2 \quad 3 \ - \ - \ i}_{(i+1)} \quad \underbrace{—— \ N-1}_{[i \text{ to } N-1]}$$

$(i+1)$

$\Downarrow$

$s \in [0-i] \Rightarrow i-0+1$

$= i+1$

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

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

$\Rightarrow \underline{\underline{[N-i]}}$

subarrays $\Rightarrow \boxed{(N-i)*(i+1)} \rightarrow \underline{\underline{\text{ans}}}.$

$$\boxed{\begin{array}{l} TC \Rightarrow O(1) \\ SC \Rightarrow O(1) \end{array}}$$

```
for(s=0; s<N; s++) {
    for(e=s; e<N; e++) {
        // [s-e] → subarray
        for(i=s; i<=e; i++) {
            print(arr[i])
        }
    }
}
```

s	e	i
0	0	1
0	1	2
0	2	3
1	1	1
1	2	2
2	2	1

$(1+2+3) + (1+2) + (1)$

$\Rightarrow \text{sum}(N) + \text{sum}(N-1) + \text{sum}(N-2)$

$$\Rightarrow \frac{N(N+1)}{2} + \frac{(N-1)(N-1+1)}{2} + \frac{(N-2)(N-2+1)}{2}$$

$$\Rightarrow \frac{N^2+N}{2} + \frac{(N^2-N)}{2} + \frac{(N^2-N-2N+2)}{2}$$

$$\Rightarrow \frac{3N^2 - 3N + 2}{2}$$

$\Rightarrow \quad \frac{3(N^2-N)+2}{2}$

$\Rightarrow \quad \boxed{3}$

$S=0 \qquad S=1 \qquad S=2$

$\mathrm{sum}(N) + \mathrm{sum}(N-1) + \mathrm{sum}(N-2 + \text{———} - \mathrm{sum}(1)$

N

$\boxed{N2} \infty \underline{\underline{N}}$