→ Subarrays Basics
→ Printing subarrays
→ Generating all subarrays
→ Max subarray sum
→ Sum of all subarray sums
→ Max sub-array sum of len = k.

Basics

: continuous part of an array is a subarray
: empty arr is not subarray
: single element is a subarray
: subarray [s - e] ⇒ len = e - s + 1

arr ⇒ 1 2 3 4 5 6 7
0 1 2 3 4 5 6

sub [3 6] ⇒ 4 5 6 7 ⇒ len = 4 ⇒ 6 - 3 + 1

ida ⇒ [3 4 5 7 8] → XX (6 is missing)

ida ⇒ [4 - 8] → ✓

Q1. Given a startIdx, endIdx print the subarray
[s] [e]

ex ⇒ arr[] ⇒

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

input(arr, 2, 4) ⇒ Op ⇒ 8 -1 2

input(arr, 0, 5) ⇒ Op ⇒ 4 6 8 -1 2 3

input (arr, 3, 3) ⇒ Op ⇒ -1

pseudo

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

TC ⇒ O(N)
SC ⇒ O(1)

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

arr[4] =

-1	3	2	3
0	1	2	3

subarrays from → 0 → 0-0 ⇒ [-1]
0-1 ⇒ [-1 3]
0-2 ⇒ [-1 3 2]
0-3 ⇒ [-1 3 2 3]
} 4

subarrays from → 1 → 1-1 ⇒ [3]
1-2 ⇒ [3 2]
1-3 ⇒ [3 2 3]
} 3

subarrays from → 2 → 2-2 ⇒ [2]
2-3 ⇒ [2 3]
} 2

subarrays from → 3 → 3-3 ⇒ [3] } 1

total ⇒ 4 + 3 + 2 + 1

⇒ 10 ⟶ sum of N natural nos.

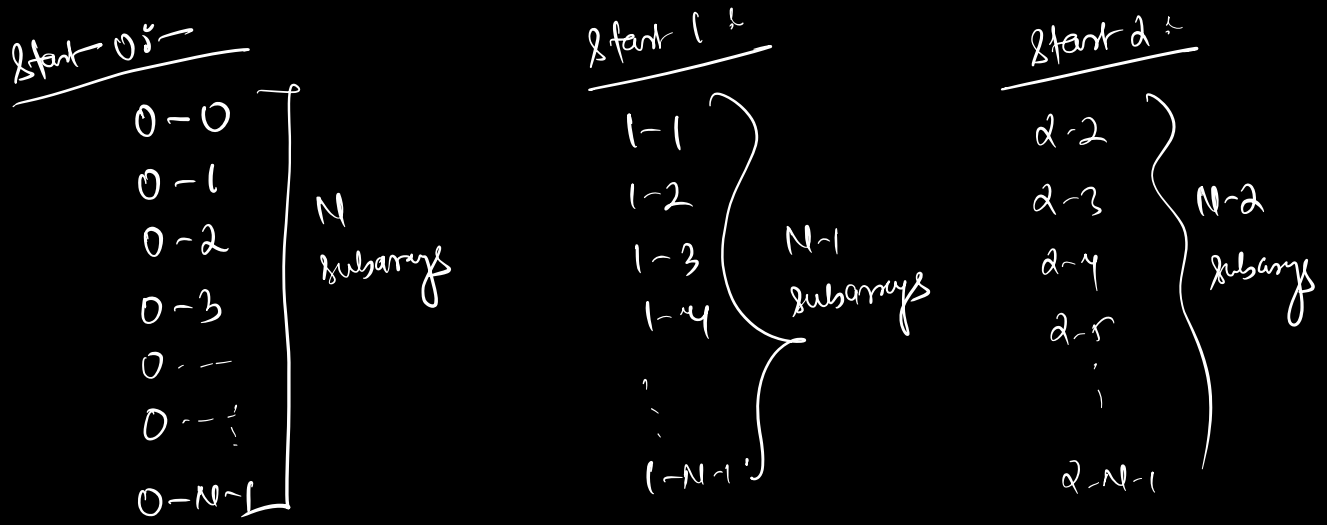
↓

4

⇒ $\frac{N(N+1)}{2} \Rightarrow \frac{4(4+1)}{2} = 10$

generalise ⇒ { 0 1 2 3 --- i --- N-2 N-1 }

N



total subarrays $\Rightarrow$ $N + N-1 + N-2 + \dots + 2 + 1$

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

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

TC $\Rightarrow$ $O(1)$

SC $\Rightarrow$ $O(1)$

Q 3. Given an array, print all subarrays.

arr[3] = [8 2 9]
0 1 2

// Start from 0 :-
i 0-0 j $\Rightarrow$ [8]
0-1 $\Rightarrow$ [8 2]
0-2 $\Rightarrow$ [8 2 9]

// Start from 1 :-
i [1] j $\Rightarrow$ 2
[1 2] j $\Rightarrow$ 2 9

// Start from 2 :-
i [2] j $\Rightarrow$ 9.

s	e
i	j

```
for(s=0; s<N; s++){
    for(e=s; e<N; e++){
        // [s-e] is subarray
        → 1) printSub(arr, s, e)
        or,
        → 2) for(i=s; i<=e; i++){
                print(arr[i])
            }
    }
}
```

TC ⇒ $O(N^3)$
SC ⇒ $O(1)$

Q 4. Given an array, print all subarray sums?

ex ⇒ arr[4] ⇒ [8 2 9 10]
0 1 2 3

0-0 → 8
0-1 → 10
0-2 → 19
0-3 → 29

1-1 → 2
1-2 → 11
1-3 → 21

2-2 → 9
2-3 → 19

3-3 → 10.

total sum = 8 + 10 + 19 + 29 + 2 + 11 + 21 + 9 + 19 + 10

= 189 + 19 + 31 + 32 + 38

= 37 + 31 + 32 + 38

= 138 ans.

Brute force

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

TC => $O(N^3)$

SC => $O(1)$

Optimisation

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

sumarr[s,e] ←

Optimised →

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

TC ⇒ $O(N + N^2) \approx O(N^2)$

SC ⇒ $O(N)$ → pefix sum array

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

eg ⇒ 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.

} O/P

approach 1 → PfSum queries

```
for( e=2; e<N; e++) {
    sum = pfsum( 2, e)
    print( sum );
}
```

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

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

2

2+1=3

3+6=9

9+8=17

17+2=19

19+3=22

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

Pseudo → carry forward → sum

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

⇒ printing all subarrays sums using carry forward :—

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

TC ⇒ $O(N^2)$

SC ⇒ $O(1)$.

Q 6. Find the maxm sum out of all subarray sums.

ex ⇒ arr ⇒ [4 (0) | -1 (1) | 2 (2) | 3 (3)]

sum = 0-0 — 4
0-1 — 3
0-2 — 5
0-3 — 8

1-1 — -1
1-2 — 1
1-3 — 4

2-2 — 2
2-3 — 5

3-3 — 3

ans = 8.

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

TC ⇒ O(N2)
SC ⇒ O(1)

→ can be solved in TC ⇒ O(N)

↓

[Kadane's Algo.]

↓

discussed during advanced lectures

Q.7. Print sum of all subarray sum:-

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

TC ⇒ $O(N^2)$

SC ⇒ $O(1)$.

Q.8. How many subarrays contains idx 3 in the given array?

arr[] ⇒

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

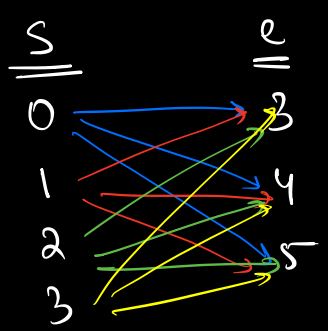
subarray →

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

Ans: 12

$[s - e] \rightarrow s \leq 3 \ \& \ e \geq 3.$

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



total no. of subarrays
⇒ no. of (s) * no. of (e)
⇒ 4 * 3
⇒ 12 subarrays

Q- How many subarrays contains idx i in any given array?

arr ⇒ { 0 1 2 3 4 --i ---- N-1 }

$s \leq i$

$[0, i] \rightarrow (i+1)$ → indexes

$(i - 0 + 1)$

$s \Rightarrow [i+1]$

$e \geq i$

$[i \quad N-1]$

$\Rightarrow N - 1 - i + 1$

$\Rightarrow (N - i)$

$e \Rightarrow [N-i]$

Total no. of subarrays $\Rightarrow \underline{(i+1)(N-i)}$

TC $\Rightarrow O(1)$

SC $\Rightarrow O(1)$

$\Rightarrow$ Sum of all subarray sum :- $\left[arr[i] * (N-i) * (i+1) \right]$ for(i=0→N-1)

arr[8 2 9]

↓

8 → 8 2 → 2 9 → 9

8 2 → 10 2 9 → 11

8 2 9 → 19

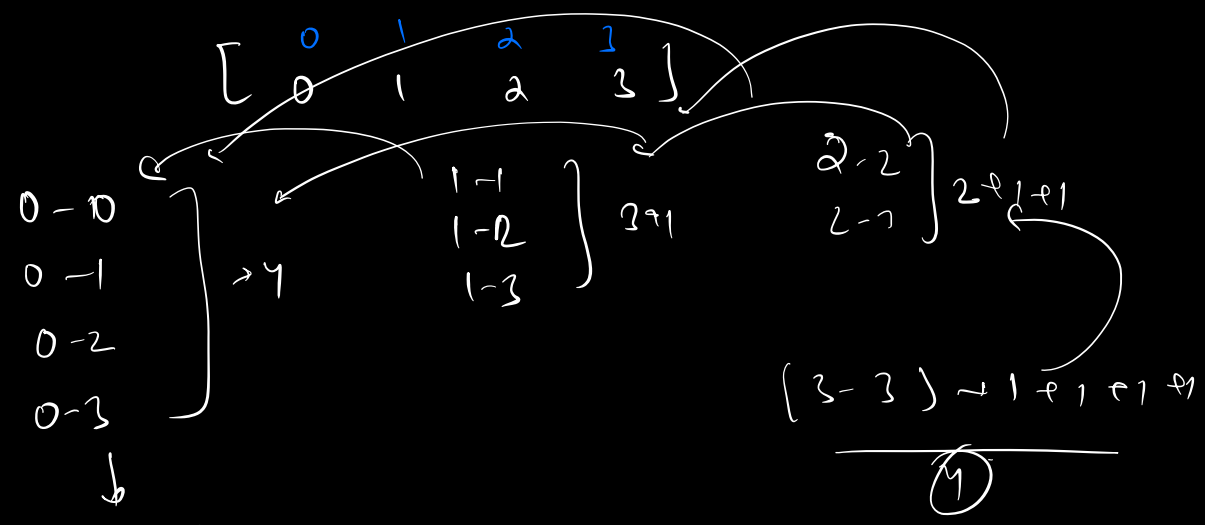
$\Rightarrow$ 59

(8) + (8+2) + (8+2+9) + (2) + (2+9) + (9)

8 * (3-0) * (1)

2 * (3-1) * (2)

9 * (3-2) * (4)



$4 * (N+1) * (i+1)$

$1 * (\quad$

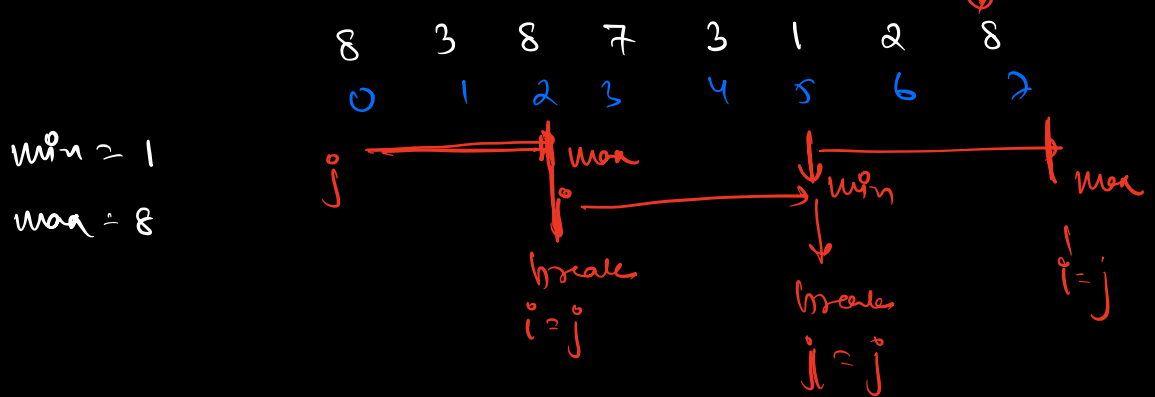
Doubt → everyone

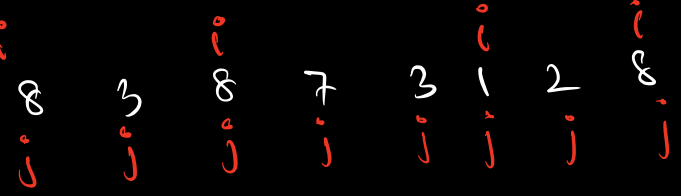
Soln → me

---

previous session discussion

1) only 1 min 2 1 max
2) min & max at bound.
3) min — max | max — min





TC = O(2N) ≈ O(N)

pseudo

1) if on a max, find min
   if on a min, find max

   keep shortest length updated.

2) i = 0, j = 0.

   iterate i till you find (min/max)

   ↓

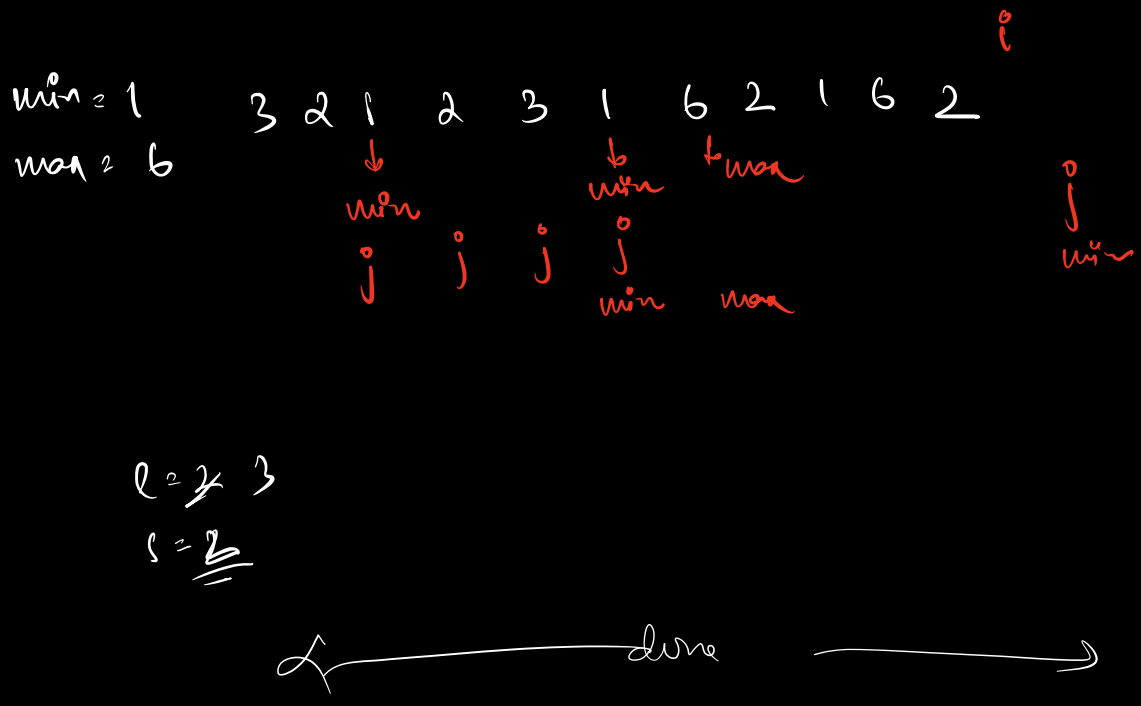
   if found a min

   iterate j till you find (min/max)

   ↓

   if find min
   → break
   → i = j

   if find max
   → len
   → shortest = min(short, len)
   → break
   → i = j



l = ~~2~~ 3

s = ~~2~~