Q→ Given an integer array, find the sum of all possible subarray sum.

$A = [2 \quad 5 \quad 3]$

	sum
2	→ 2
2 5	→ 7
2 5 3	→ 10
5	→ 5
5 3	→ 8
3	→ 3
	35 (Ans)

Bruteforce →

```
ans = 0
for L → 0 to (N-1) {
    for R → L to (N-1) {        // L — R
        sum = 0                 → subarray sum
        for i → L to R {        from L to R
            sum += A[i]
        }
        ans += sum
    }
}
return ans
```

TC = $O(N^3)$

SC = $O(1)$

L	R	
0	0	
0	1	
0	2	
0	3	A[0] + A[1] + A[2] + A[3]
0	4	A[0] + A[1] + A[2] + A[3] + A[4]
⋮	⋮	

```
ans = 0
for L → 0 to (N-1) {
    sum = 0
    for R → L to (N-1) {   // L — R
        sum += A[R]        // calculate
        ans += sum         // use        } carry forward
                ↓
            max/min/gcd...
    }
} return ans
```

TC = O(N²) SC = O(1)

A = [2 5 3]
    0 1 2

L	R	sum	ans
0	0	2	2
	1	2+5=7	2+7=9
	2	7+3=10	9+10=19
1	1	5	19+5=24
	2	5+3=8	24+8=32
2	2	3	32+3=35 (Ans)

A = [2 5 3]
    0 1 2

Sum

2 → 2

2 5 → 7

2 5 3 → 10

5 → 5

5 3 → 8

3 → 3

35 (Ans)

Contribution Technique

use when one element is used multiple times to calculate the answer.

2*3 + 5*4 + 3*3 = 35

Ans = $\sum_{\forall i}$ contribution of A[i]

A[i] * (# subarrays A[i] is a part of)

How to find # subarrays where A[i] is present?

```
          i
    0   1   2   3   4   5
A = [ 3  -2   4  -1   2   6 ]
```

subarray $\rightarrow (i+1)*(N-i)$

start $\rightarrow$ {0, 1} [0 i] $\rightarrow$ i+1

end $\rightarrow$ {1, 2, 3, 4, 5} [i N-1] $\rightarrow$ N-1-i+1 = N-i

2 * 5 = 10

```
          i
    0   1   2   3   4   5
A = [ 3  -2   4  -1   2   6 ]
```

subarray = (2+1) * (6-2) = 3 * 4 = 12

$$Ans = \sum_{\forall i} A[i] * (i+1) * (N-i)$$

```
ans = 0
for i ⟶ 0 to (N-1) {
    ans += A[i] * (i+1) * (N-i)
}
return ans
```

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

```
    0   1   2   3   4   5
A = [ 3  -2   4  -1   2   6 ]
```

N = 6

K = 4 → # subarrays of length K = 3

N = 7 K = 4

0 — 6

L	R
0	3 → (K-1)
1	4
2	5

3 | 6 → (N-1)

subarrays of length K = [(K-1) (N-1)]

= (N-1) - (K-1) + 1

= N - K + 1 ✓

Q → Given an integer array, print the start & end index of all subarrays of length K.

N = 7 K = 4

L R

o/p → 0 3

1 4

2 5

3 6 → (N-1)

[L (N-1)] = (N-1) - L + 1 = K

⇒ N - K = L

```
for L → 0 to (N-K) {
    R = L + (K-1)
    print (L, R)
}
```

[L R] = R - L + 1 = K

⇒ R = K + L - 1 = L + K - 1

Q → Given an integer array, print the max subarray sum of length K.

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

K = 4

4 3 11

Ans = 11

Bruteforce →

```
ans = 0
for L → 0 to (N-K) {
    R = L + (K-1)
    sum = 0
    for i → L to R {
        sum += A[i]
    }
    ans = max(ans, sum)
}
return ans
```

TC → $O((N-K+1)*K)$

$K = 1 \rightarrow (N-1+1)*1 = N$

$K = N \rightarrow (N-N+1)*N = N$

→ length K subarray sum

$K = N/2 \quad (N-\frac{N}{2}+1)*\frac{N}{2}$

$= (\frac{N}{2}+1)*\frac{N}{2} \rightarrow O(N^2)$

worst case TC = $O(N^2)$

Find subarray sum from index L to R ↘

(prefix sum) P[R] - P[L-1]

```
P[0] = A[0]
for i → 1 to (N-1) {
    P[i] = P[i-1] + A[i]
}
ans = P[K-1]    // 0—(K-1)
for L → 1 to (N-K) {
    R = L + (K-1)
    ans = max(ans, P[R] - P[L-1])
}
```

✓

* index out of bound error

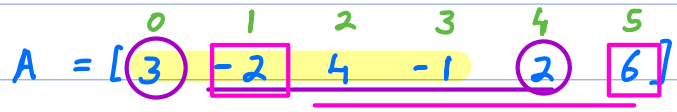
return ans

TC = O(N+N) = O(N)

SC = O(N) P[] → use A to store prefix sum

SC = O(1)

SC = O(1) without updating input → carray forward



K = 4

sum = 4 + 2 − 3 = 3 + 6 − (−2) = 11

```
sum = 0
for i → 0 to (K-1) {
    sum += A[i]
}
```

Sliding Window

```
ans = sum
for i → K to (N-1) {          // [L  R] → K
    sum += A[i]                 L  i
    sum -= A[i-K]             // L-1 = i-K
    ans = max(ans, sum)
}
```

K ⇒ i − L + 1 = K

⇒ L = i − K + 1

return ans TC = O(N) SC = O(1)

---

subarray
prefix sum
carry forward
sliding window
contribution technique

Lectures → understand concept

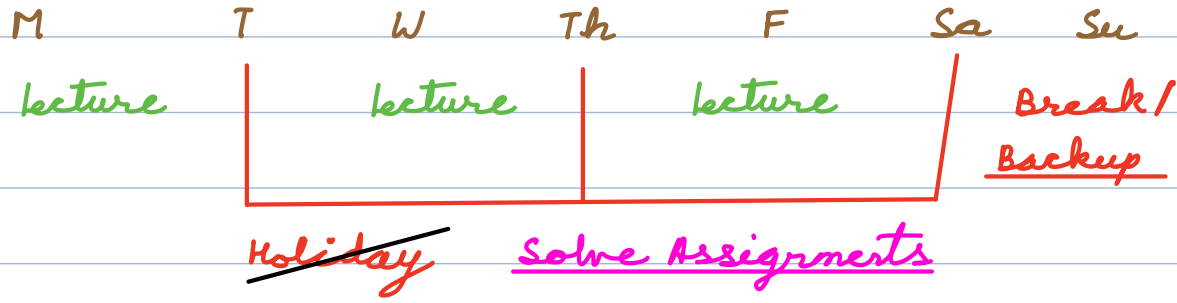
Assignments → retain concepts

V.V.V.... important

short term → contest

long term → Mock Interview

} Actual interview



Daily → complete all Non-Scaler work before 9:00 PM.

9 — 12 (night) → Scaler Tasks

21 days ⇒ Habit