

Q → Given an integer array, find the sum of all possible subarray (continuous part of array).

$A = [1 \ 2 \ 3]$

1	→	1
1 2	→	3
1 2 3	→	6
2	→	2
2 3	→	5
3	→	3

subarrays
 $= N * (N+1) / 2$

Bruteforce →

```
for i → 0 to (N-1) {  
  for j → i to (N-1) { // i → j  
    sum = 0  
    for k → i to j {  
      sum += A[k]  
    }  
    print (sum)  
  }  
}
```

$TC = \underline{O(N^3)}$ $SC = \underline{O(1)}$

Prefix Sum

$P[i] = A[0] + A[1] + \dots + A[i]$

$P[i] = P[i-1] + A[i]$

subarray sum from index i to j → $P[j] - P[i-1]$, $i > 0$
→ $P[j]$, $i == 0$

```

P[0] = A[0]
for i → 1 to (N-1) {
    P[i] = P[i-1] + A[i]
}

```

```

for i → 0 to (N-1) {
    for j → i to (N-1) { // i → j
        if (i == 0) sum = P[j]
        else sum = P[j] - P[i-1]
        print (sum)
    }
}

```

TC = $O(N + N^2) = \underline{O(N^2)}$ ✓

SC = $\underline{O(N)}$ → use A to store
prefix sum → $\underline{O(1)}$

A = [1⁰ 2¹ 3²]

P = [1 3 6]

1	→	1	✓
1 2	→	3	✓
1 2 3	→	6	✓
2	→	2	✓
2 3	→	5	✓
3	→	3	✓

Carry Forward (calculate + use)

A = [1⁰ 2¹ 3²]

```

for i → 0 to (N-1) {
    sum = 0
    for j → i to (N-1) { // i → j
        sum += A[j] // calculate
        print (sum) // use
    }
}

```

previous sum + last element

i	j	sum
0	0	1
	1	3
	2	6
1	1	2
	2	5
2	2	3

$$TC = O(N^2) \quad SC = O(1)$$

Q → Find the total sum of all subarray sums.

ans = 0

```

for i → 0 to (N-1) {
    sum = 0
    for j → i to (N-1) { // i → j
        sum += A[j]
        ans += sum
    }
} return ans

```

$A = [1, 2, 3]$

1	→	1
1 2	→	3
1 2 3	→	6
2	→	2
2 3	→	5
3	→	3

Ans = 20

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

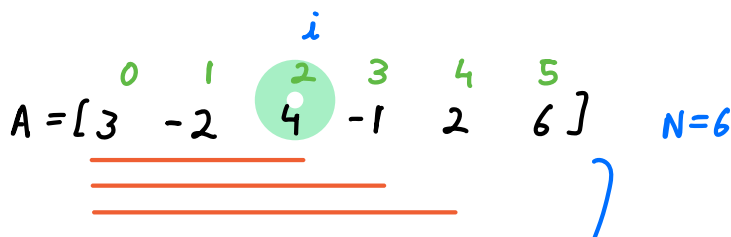
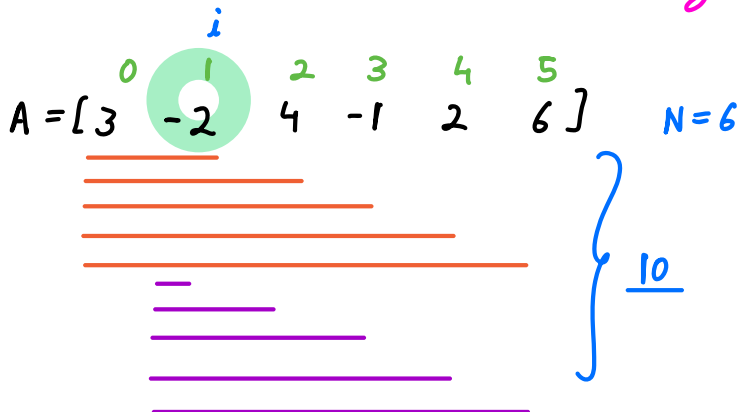
$$(1 \times 3 + 2 \times 4 + 3 \times 3)$$

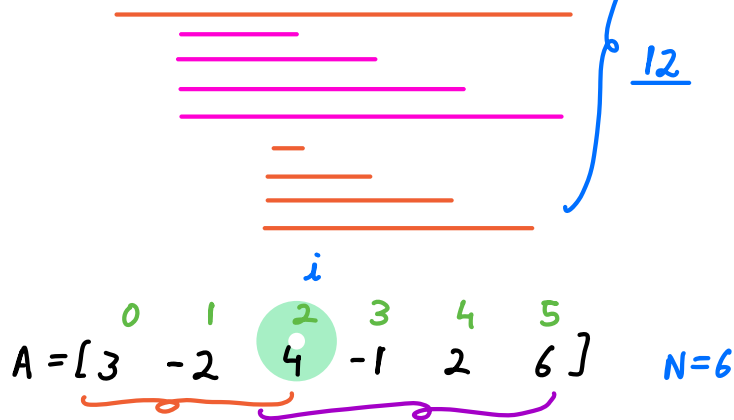
$$3 + 8 + 9 = 20$$

Contribution Technique

$$Ans = \sum_{i=0}^{N-1} (\text{contribution of } A[i])$$

$A[i] \times (\# \text{ subarrays } A[i] \text{ is a part of})$





start index can be $\rightarrow [0 \quad i] \rightarrow (i+1)$

end index can be $\rightarrow [i \quad N-1] \rightarrow N-1-i+1 = (N-i)$

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

ans = 0

for $i \rightarrow 0$ to $(N-1)$ {

 ans += $A[i] * (i+1) * (N-i)$

} return ans

TC = $O(N)$

SC = $O(1)$

10:35 PM

Q \rightarrow Total # subarrays of length K ($\leq N$) = ?

$A = [3 \quad -2 \quad 4 \quad -1 \quad 2 \quad 6]$ $K=3$

 Ans = 4

$A = [3 \quad -2 \quad 4 \quad -1 \quad 2 \quad 6]$

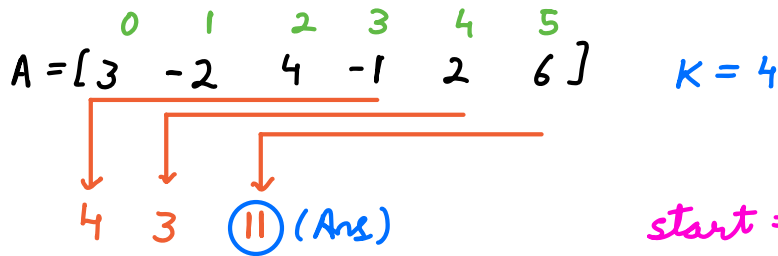
<u>K</u>	<u># subarrays</u>
1	6 (N)
2	5 (N-1)
3	4 (N-2)
4	3 :
5	2 :
6	1 (N-N+1)

subarray of length K
= $N - K + 1$

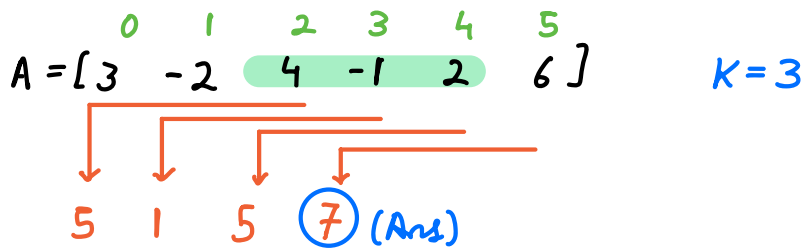
$N=7 \quad K=4$

subarrays = $7 - 4 + 1 = \underline{4}$

Q → Given an integer array, find max subarray sum of subarray with length = K.



start = 0 len = K \Rightarrow end = K-1



```

st = 0    end = K-1    ans = INT_MIN
→ while (end < N) {
    sum = 0
    for i → st to end {
        sum += A[i]
    }
    ans = max(ans, sum)
    st++    end++    // sliding window
}
return ans

```

$[K-1 \quad N-1] \rightarrow N-K+1$
 $\rightarrow K$
 $\frac{K}{1} \rightarrow (N-K+1) * K$
 $N \rightarrow N * 1 = N$
 $\frac{N}{2} \rightarrow (N - \frac{N}{2} + 1) * \frac{N}{2}$
 $\approx \frac{N^2}{4} \rightarrow O(N^2)$

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

Prefix Sum

```

st = 0    end = K-1    ans = INT_MIN
while (end < N) {

```

```

if (st == 0) sum = P[er]
else sum = P[er] - P[st-1]

```

```
ans = max(ans, sum)
```

```
st++ end++ // sliding window
```

```
}
```

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

```
return ans
```

$SC = O(N)$ → use A[] to store P[] → $SC = O(1)$

Carry Forward
(Sliding Window)

$A = [3, -2, 4, -1, 2, 6]$ $K = 4$
 Indices: 0 1 2 3 4 5
 (K-1) K
 A window of size 4 is shown from index 0 to 3, and another window from index 3 to 6.

```
sum = 0
```

```
for i → 0 to (K-1) { // 0 — K-1
```

```
    sum += A[i]
```

```
}
```

```
ans = sum
```

```
for i → K to (N-1) { // — end = i
```

```
    sum += A[i]
```

```
    sum -= A[i-K]
```

```
    ans = max(ans, sum)
```

```
}
```

```
return ans
```

$TC = O(N)$

$SC = O(1)$