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

$\boxed{\# \text{ subarrays} = N*(N+1)/2}$

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 = $O(N^3)$ SC = $O(1)$

Prefix Sum

$P[i] = A[0] + A[1] + ... + A[i]$

$\boxed{P[i] = P[i-1] + A[i]}$

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 → $\underline{O(1)}$

$$A = \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \end{array}$$

$$P = \begin{bmatrix} 1 & 3 & 6 \end{bmatrix}$$

i
j

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

Carry Forward (calculate + use)

$$A = \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \end{array}$$

```
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)   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
```

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

$$A = \begin{array}{ccc} 0 & 1 & 2 \\ [1 & 2 & 3] \end{array}$$

1 → 1

1 2 → 3

1 2 3 → 6

2 → 2

2 3 → 5

3 → 3

Ans = 20

(1*3 + 2*4 + 3*3)

3 + 8 + 9 = 20

Contribution Technique

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

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

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ [3 & -2 & 4 & -1 & 2 & 6] \end{array} \quad N=6$$

i = 1

10

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ [3 & -2 & 4 & -1 & 2 & 6] \end{array} \quad N=6$$

i = 2

12

i

0	1	2	3	4	5

A = [3 -2 4 -1 2 6]   N=6

start index can be → [0 i] → (i+1)

end index can be → [i N-1] → N-1 - i + 1 = (N-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)

10:35 PM

Q → Total # subarrays of length K (<= N) = ?

0	1	2	3	4	5

A = [3 -2 4 -1 2 6]   K=3

Ans = 4

0	1	2	3	4	5

A = [3 -2 4 -1 2 6]

K	# subarrays	
1	6	(N)
2	5	(N-1)
3	4	(N-2)
4	3	⋮
5	2	⋮
6	1	(N-N+1)

$\boxed{\text{\# subarray of length K} = N-K+1}$

N=7   K=4

subarrays = 7 - 4 + 1 = 4

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

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

(indices 0 1 2 3 4 5)

4 3 11 (Ans)

start = 0   len = K ⇒ end = K-1

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

(indices 0 1 2 3 4 5)

5 1 5 7 (Ans)

```
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 N-1] → N-K+1

K → (N-K+1) * K

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

$\approx \frac{N^2}{4} \rightarrow O(N^2)$

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[end]
    else   sum = P[end] - P[st-1]

    ans = max(ans, sum)
    st++    end++    // sliding window
}
return ans
```

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

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

Carry Forward
(Sliding Window)

	0	1	2	3 (K-1)	4 (K)	5
A =	3	-2	4	-1	2	6

K = 4

```
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 = $\underline{O(N)}$ SC = $\underline{O(1)}$