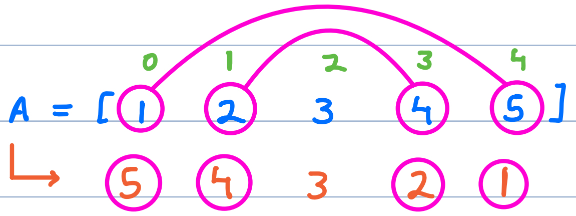
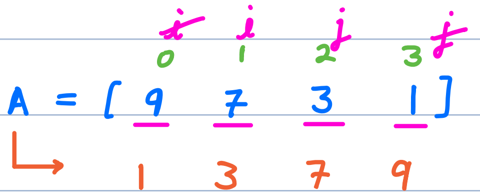
Q → Reverse the given integer array.





```
i = 0        j = N-1
while (i < j) {
    swap (A, i, j)  →  t = A[i]
    i++                A[i] = A[j]
    j--                A[j] = t
}
```

swap (x, y)

x = 7

y = 4

x = x + y (7 + 4)

y = x − y (7)

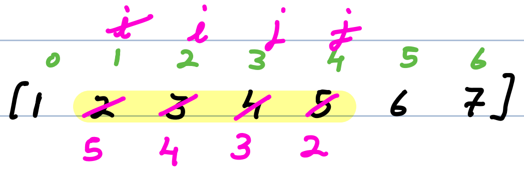
x = x − y

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

Q → Reverse a subarray from index L to R.

subarray → continuous part of array

```
i = L        j = R
while (i < j) {
    swap (A, i, j)
    i++
    j--
}
```



L = 1

R = 4

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

Q → Given an integer array & multiple queries, for each query find sum of elements from L to R.

Perform a task multiple times.

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

Queries

(L ,	R)	
2	5	→ 13
0	1	→ 3
4	6	→ 15

// B [N][2] → B[i][0] } i^{th} query
B [i][1] }

Bruteforce →

```
for i → 0 to (Q-1) {
    L = B[i][0]      R = B[i][1]
    sum = 0
    for j → L to R {
        sum += A[j]
    }
    print (sum)
}
```

TC = O(Q * N)

SC = O(1)

$Q, N <= 10^5$

⇒ TLE

Cricket (Harsh)

Over →		1	2	3	4	5	6	7	8	9	10
Score →	0	2	8	14	29	31	49	65	79	88	97

after every over

Runs scored in 7^{th} over = 65 - 49 = 16

Runs scored from 6^{th} to 10^{th} over = 97 - 31 = 66

Runs scored in 10^{th} over = 97 - 88 = 9

Runs scored from 3^{rd} to 6^{th} over = 49 - 8 = 41

Runs scored from 4th to 9th over = 88 - 14 = $\underline{74}$

Prefix sum

$P[i] = A[0] + A[1] + A[2] + \dots + A[i]$

$$\boxed{\begin{array}{l} P[0] = A[0] \\ P[i] = P[i-1] + A[i] \end{array}}$$

	0	1	2	3	4	5
A =	10	32	6	12	20	1
P =	10	42	48	60	80	81

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

Queries

	0	1	2	3	4	5
A =	10	32	6	12	20	1
P =	10	42	48	60	80	81

1 5 → 81 - 10 = 71

$$\boxed{L \quad R \quad \longrightarrow \quad P[R] - P[L-1]}$$

0 2 → P[2] = 48

```
for i → 0 to (Q-1) {
    L = B[i][0]        R = B[i][1]
    if (L > 0)     print (P[R] - P[L-1])
    else           print (P[R]                TC = O(Q)
}
```

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

SC = $\underline{O(N)}$ → P[]

$\underline{O(1)}$ (update A[] to prefix sum)

~~$A[0] = A[0]$~~

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

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ [10 & \cancel{32} & \cancel{6} & \cancel{12} & \cancel{20} & \cancel{1}] \\ & 42 & 48 & 60 & 80 & 81 \end{array}$$

Q → Given an integer array, find the count of equilibrium index i.e an index 'i' s.t

sum of elements on left of i =

sum of elements on right of i.

$$A = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ [-3 & 2 & 4 & -1] \end{array}$$

i	Left	Right	
0	0	5	✗
1	-3	3	✗
2	-1	-1	✓
3	3	0	✗

Ans = 1

$$A = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ [\underline{-7} & \underline{1} & \underline{5} & 2 & \underline{-4} & \underline{3} & \underline{0}] \end{array}$$

i	Left	Right	
3	-1	-1	✓
6	0	0	✓

Ans = 2

Bruteforce → ∀i, check if its eq. index.

TC = $O(N^2)$

Sum $(0 — (i-1))$ → $P[i-1]$

Sum $((i+1) — (N-1))$ → $P[N-1] - P[i]$

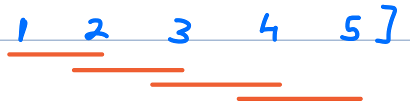
Prefix Sum →

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

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

Q → How many subarray of length K are present in array.

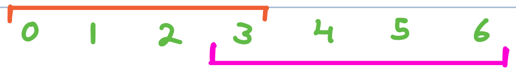
$A = [1 \quad 2 \quad 3 \quad 4 \quad 5]$ $\quad K = 2 \quad Ans = 4$



$K = 5 \quad Ans = 1$

$N = 7$

$K = 4$



First subarray → start = 0

end = K-1

$[L \quad R] \rightarrow \underline{R - L + 1}$

Last subarray → start = N-K

end = N-1

$[x \quad (N-1)] \rightarrow (N-1) - x + 1 = K$

$\Rightarrow \underline{N - K = x}$

subarrays → $[0 \quad N-K] \rightarrow N-K-0+1 = \underline{N-K+1}$

of len K $\quad [K-1 \quad N-1] \rightarrow N-1-(K-1)+1 = \underline{N-K+1}$ ✓

Q → Print start & end index of every subarray.

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

```
for e → (K-1) to (N-1) {
    s = e - K + 1
    print (s, e)
}
```

Q → Given an integer array, find max subarray sum of len K subarray.

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = [& -3 & 4 & -2 & 5 & 3 & -2 & 8\ ] \end{array} \qquad K = 5$$

7 8 12   Ans = 12

Bruteforce → ∀ subarray of len K, travel & calculate sum.

$TC = O((N-K+1) * K)$

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

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

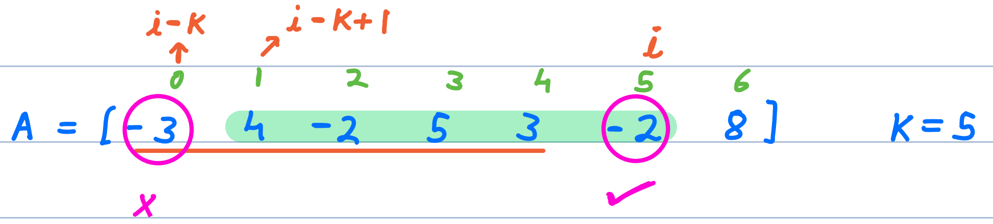
$K = \frac{N}{2} \rightarrow (N - \frac{N}{2} + 1) * \frac{N}{2} \approx \frac{N^2}{4} \rightarrow \underline{O(N^2)}$

Sol → ∀ subarray of len K, calculate sum.

```
ans = INT_MIN
for e → (K-1) to (N-1) {
    s = e - K + 1
    // Prefix sum
    if ( s > 0 )    sum = P[e] - P[s-1]
    else    sum = P[e]
    ans = max (ans, sum)
}
```

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

Without updating A[] → O(1) ? ✓



Sliding Window

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

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

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

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

A = [2 5 3]

2 → 2
2 5 → 7
2 5 3 → 10
5 → 5
5 3 → 8
3 → 3

35 (Ans)

```
ans = 0
for s → 0 to (N-1) {
|    for e → s to (N-1) {    // s — e
|    |    sum = 0
```

```
        for i → s to e {
            sum += A[i]
        }                      → P[e] - P[s-1]
        ans += sum
    }
} return ans
```

Prefix Sum

TC = $\underline{O(N^3)}$ → (N^2)

SC = $\underline{O(1)}$ → $O(N)$

Carry Forward

```
ans = 0
for s → 0 to (N-1) {
    sum = 0
    for e → s to (N-1) {   // s — e
        sum += A[e]
        ans + = sum
    }
} return ans
```

~~s~~ ~~s~~ s

0 1 2

A = [2 5 3]

e

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

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

sum = ~~0~~ 3

ans = 0 + 2 + 7 + 10
+ 5 + 8 + 3 = $\underline{35}$

Contribution Technique

→ 1) One element is used multiple Times in answer calculation ✓

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

A = [2 5 3]

2 → 2

2 5 → 7

2 5 3 → 10

5 → 5

5 3 → 8

3 → 3

35 (Ans)

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

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

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 3 & \boxed{-2} & 4 & -1 & 2 & 6 \end{array}$$

start → [0 1] → 2

end → [1 5] → 5

$> 2*5 = \underline{10}$

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 3 & -2 & \boxed{4} & -1 & 2 & 6 \end{array}$$

start → [0 2] → 3

end → [2 5] → 4

$>$ 12

subarrays where A[i] is present =

start → [0 i] → $i+1$

end → [i (N-1)] → $N-1-i+1 = N-i$

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

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

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

```
for (i = N/2 ; i <= N ; i++) → N/2
    for (j = 2 ; j <= N ; j = j*2) → log (N)
```

$2 \rightarrow 4 \rightarrow \ldots \; 2^K = N$

$K = \log(N)$

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