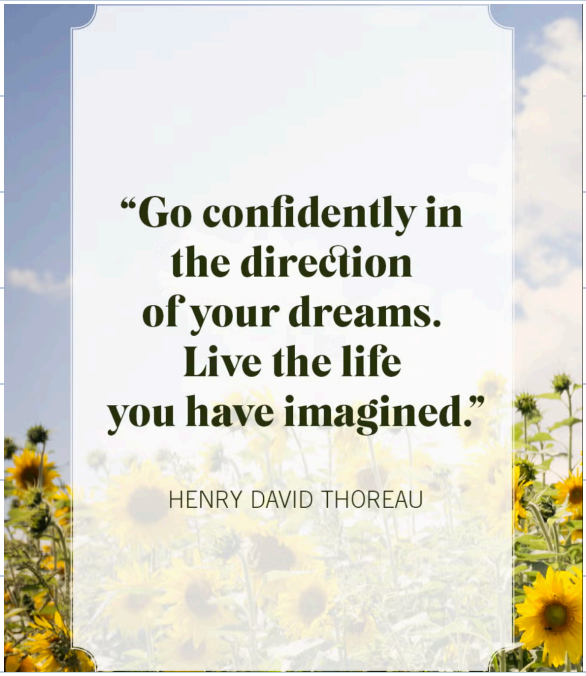
Agenda:

1. Count of Pairs ag - [Carry Forward Technique]
2. Introduction to Subarrays
3. Find sum of all Subarrays sums - [Contribution Technique]
4. Max subarray sum with length K - [Sliding Window Technique]



Q → Given a string of lower case letters, return the count of i, j s.t i < j & s[i] = 'a' & s[j] = 'g'.

" 0 1 2 3 4 5 6 "
s = a b e g a c g

i, j
0, 3
0, 6
4, 6

Ans = 3

s = "a c g d g a g"
0 1 2 3 4 5 6

(0, 2) (0, 4)

(0, 6) (5, 6) Ans = 4

s = "b c a g g a a g"
0 1 2 3 4 5 6 7

(2, 3) (2, 4) (2, 7)

(5, 7) (6, 7) Ans = 5

Bruteforce

```
cnt = 0
for i → 0 to (N-1) {
    for j → 0 to (N-1) {
        if ( i < j && s[i] == 'a' && s[j] == 'g')
            cnt ++
    }
}
} return cnt
```

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

```
cnt = 0
for i → 0 to (N-1) {
    if (s[i] == 'a') {
        for j → i+1 to (N-1) {
            if (s[j] == 'g')  cnt++
        }
    }
} return cnt
```

→ count # of 'g' from (i+1) till end.

TC = $\underline{O(N^2)}$ SC = $\underline{O(1)}$

⇒ count # of 'g' from ~~(i+1)~~ ∀ index till end.

	0	1	2	3	4	5	6	7	8
s =	a	c	g	b	g	k	a	g	c
→	3	3	3	2	2	1	1	1	0

```
∀ i, cnt_g[i] = 0
if (s[N-1] == 'g')  cnt_g[N-1] = 1
for i → N-2 to 0 {
    if (s[i] == 'g')  cnt_g[i] = cnt_g[i+1] + 1
    else              cnt_g[i] = cnt_g[i+1]
}
```

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

```
ans = 0        cnt = 0
for i → (N-1) to 0 {
    if (s[i] == 'g')  cnt++      // calculate  } carry Forward
    if (s[i] == 'a')  ans += cnt // use        }
}
return ans
```

```
      i  i  i  i  i  i  i  i  i
      0  1  2  3  4  5  6  7  8
s =   a  c  g  a  g  k  a  g  c
```

cnt = 0 + 2 3

ans = 0 + 3 6

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

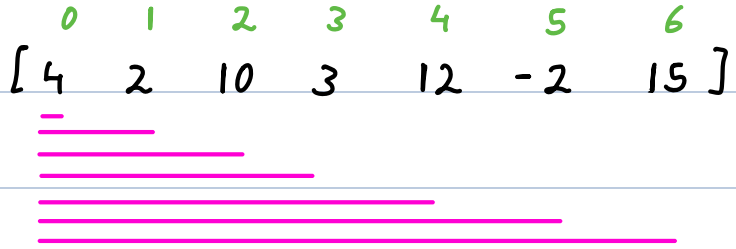
Subarrays → continuous part of array.

```
      0  1  2  3  4  5
[     4  2  1  6  8  3 ]
```

```
      0  1  2  3  4  5  6  7
[     2  4  1  6 -3  7  8  4 ]
```

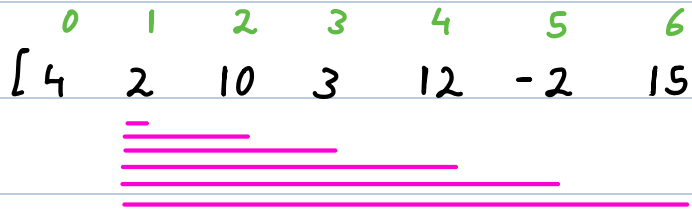
✓ start — end → (5, 7)

start, length → 5, 3



Ans = 7

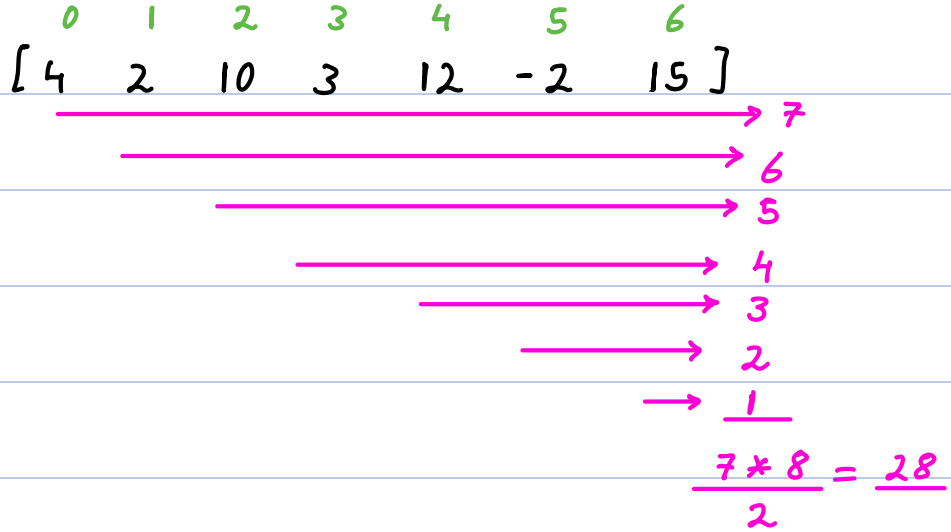
end → [0 6] → 6 - 0 + 1 = 7



Ans = 6

end → [1 6] → 6 - 1 + 1 = 6

What is total # subarrays in an array of len = N ?



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

$\frac{7 * 8}{2} = 28$

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

A = [3 2 5]

3 → 3

3 2 → 5

3 2 5 → 10

2 → 2

2 5 → 7

5 → 5

32 (Ans)

Bruteforce →

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

TC = $O(N^3)$

SC = $O(1)$

Prefix Sum

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

```
ans = 0
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]
        ans += sum
    }
} return ans
```

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

Carry Forward

```
ans = 0
for i → 0 to (N-1) {
    sum = 0                          // i = 0
    for j → i to (N-1) {             // i — j
        sum += A[j]                  // A[0] → A[0]+A[1] → A[0]+A[1]+A[2]
        ans += sum
    }
}
return ans
```

A = [3 2 5] (indices 0 1 2)

i	j	sum	ans
0		0	0
	0	3	3
	1	5	3 + 5 = 8
	2	10	8 + 10 = 18
1		0	
	1	2	18 + 2 = 20
	2	7	20 + 7 = 27
2		0	
	2	5	27 + 5 = 32

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

Contribution Technique

If one element is contributing multiple times in answer then →

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

A = [3 2 5]

3 → 3

3 2 → 5

3 2 5 → 10

2 → 2

2 5 → 7

5 → 5

32 (Ans)

3*3 + 2*4 + 5*3

= 32

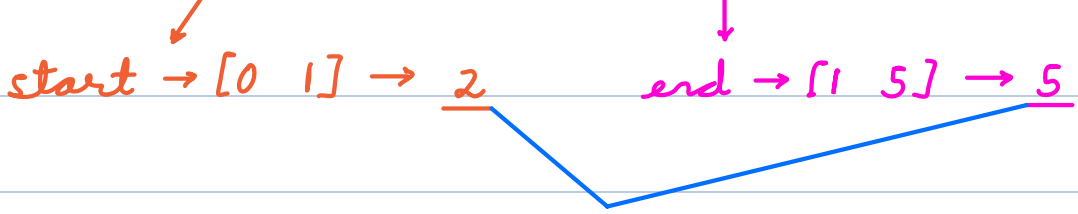
$$ans = \sum_{\forall i} \text{contribution of } A[i]$$

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

Find # subarrays where index 1 is present.

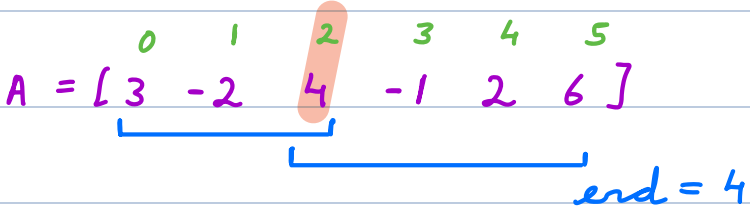
0	1	2	3	4	5
3	-2	4	-1	2	6

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



subarrays = 2 * 5 = 10

Find # subarrays where index 2 is present.



start → 3 # subarrays = 3 * 4 = 12

i → start [0 i] → i - 0 + 1 = i + 1

end [i N-1] → N - 1 - i + 1 = N - i

contribution of A[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)

Q → What is the count of subarrays of length K.

0	1	2	3	4	5	6
4	2	10	3	12	-2	15

K = 3

Ans = 5

K = 4

0	1	2	3	4	5	6
4	2	10	3	12	-2	15

Ans = 4

N, K

End → first subarray = K-1

last subarray = N-1

[K-1 N-1] → N-1-(K-1)+1 = N-K+1

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

A =

0	1	2	3	4	5	6
4	2	10	3	12	-2	15

K = 3

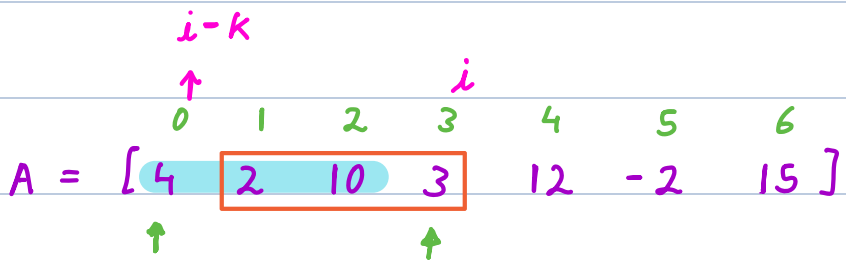
16 15 25 13 25

Ans = 25

Bruteforce →

```
ans = 0  INT_MIN
i = 0      j = K-1
while (j < N) {
    sum = 0
    for k → i to j {
        sum += A[k]
    }
    ans = max(ans, sum)
    i++    j++
}
```

TC = O(N²) SC = O(1)



K = 3

i	j	sum	ans
0	2	16	16
1	3	16 + 3 - 4 = 15	16
2	4	15 + 12 - 2 = 25	25
3	5	25 + (-2) - 10 = 13	25
4	6	13 + 15 - 3 = 25	25

// bruteforce

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

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

return ans
```

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