Def. of a subarray.

continuous sequence of an array

1) Full array is a subarray ✓
2) Single ele is also a subarray ✓

A[] = { -3 4 5 -6 8 9 10 -10 8 }
0 1 2 3 4 5 6 7 8

0-5 ✓
2-3, 6-8 ✗
5-5 ✓
0-8 ✓

si <= ei

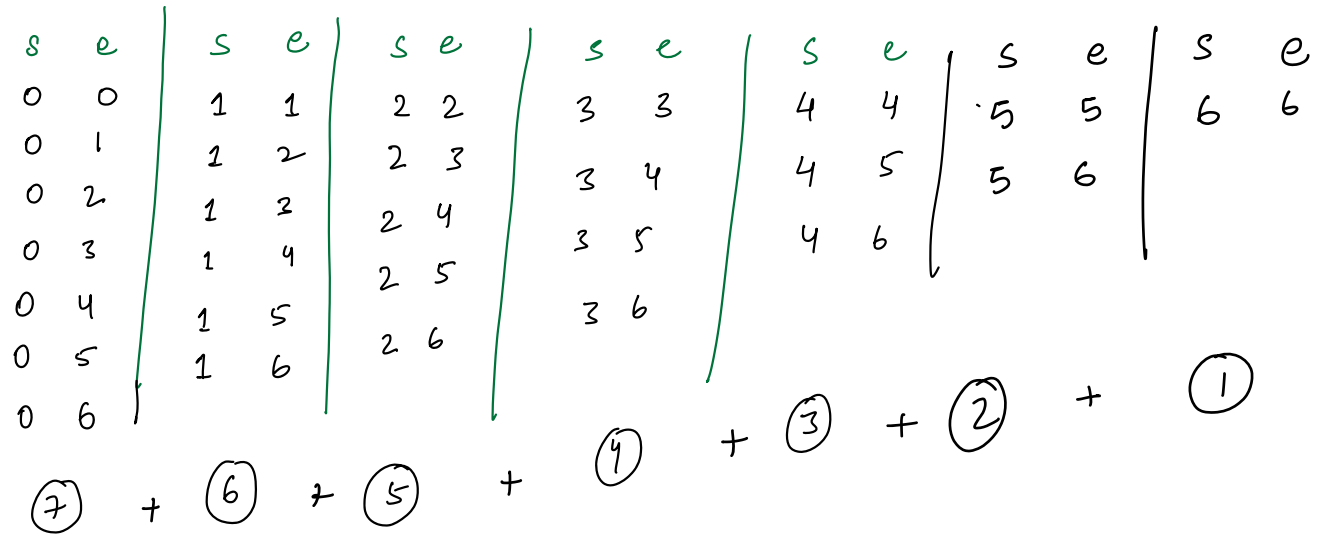
[4, 5, 1, 9, 0, 2, 3, 5]

[5] ✓
[4,5,1,0] ✗
[9,0,2,3] ✓
[4,5,1] ✓

Total no. of subarrays

ar[]: { 3 4 5 6 -2 8 10 }
0 1 2 3 4 5 6

Can a subarray start at index 0? s<=e



$$1+2+3+4+\cdots+n$$

$$\frac{n(n+1)}{2}$$

Generate all subarrays

A subarray can start index $0, 1, 2, \ldots n-1$

```
for(i=0 ; i<n; i++)
```

$i=0 \quad j=[0 \quad n-1]$

$i=1 \quad j=[1 \quad n-1]$

$i=2 \quad j=[2 \quad n-1]$

$\vdots \qquad \vdots$

```
for (i = 0; i < n; i++) {
    for (j = i; j < n; j++) {
        sum = 0
        for (k = i; k <= j; k++) {
            sum = sum + A[k]
        }
        print (sum)
    }
}
```

Que. Maximum Subarray Sum.

$a[] = \{4 \quad 2 \quad -1 \quad 3\}$

JP Morgan
LinkedIn
Service Now
Walmart
Cisco
VMware
Goldman Sachs.

[4] → 4
[4,2] → 6
[4,2,-1] → 5
[4,2,-1,3] → 8

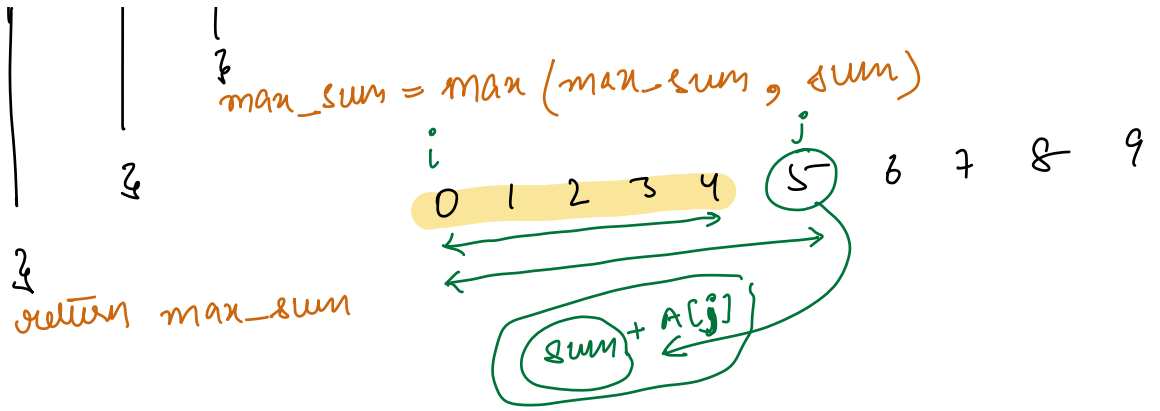
[2] → 2
[2,-1] → 1
[2,-1,3]
↳ 4

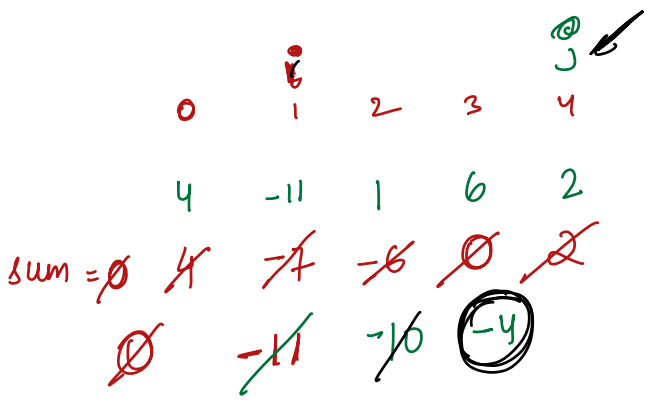
[-1] → -1
[-1,3] → 2
[3] → 3

max_sum = $-\infty$

```
for (i = 0; i < n; i++) {
    for (j = i; j < n; j++) {
        sum = 0
        for (k = i; k <= j; k++) {
            sum = sum + A[k]
```

TC
$O(n^3)$





max_sum
= -∞
4

```
max_sum = -∞
for( i = 0; i < n; i++ ) {
    sum = 0
    for( j = i; j < n; j++ ) {
        sum = sum + A[j]
        max_sum = max (max_sum, sum)
    }
}
return max_sum
```

TC
$O(n^2)$

SC
$O(1)$

Technique
Carry forward

Prefix Sum

① Create prefix sum array

② `for(i=0; i<n; i++) {`

`for(j=i; j<n; j++) {`

`[i j] // O(1)` `pf[j]-pf[i-1]`

`}`

`}`

TC

$O(n^2)$

SC

$O(n)$

⇓

Carry Forward OR Prefix Array ?

$O(n^3) \longrightarrow O(n^2) \longrightarrow O(n)$?

↓

Kadane's. → Advance

n = length of array | 0 1 2 3 4 5 6

No. of subarrays of length K.

K

K = 1

length 1	length 2	length 3	length 4
0 0	0 1	0 2	0 3
1 1	1 2	1 3	1 4
2 2	2 3	2 4	2 5
3 3	3 4	3 5	3 6
4 4	4 5	4 6	
5 5	5 6		
6 6			

$7-1+1$ | $7-2+1 = 6$ | $7-3+1 = 5$ | $7-4+1 = 4$

* K, each start will have a unique end pt. ?

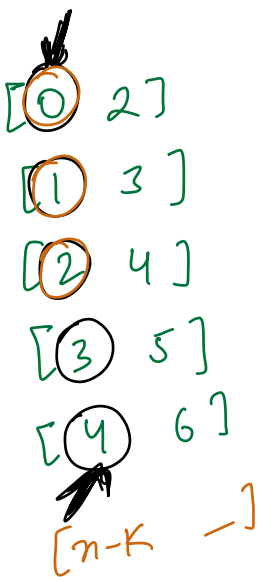
If I count just the starting pts. of subarray of size K, then I can get the total no. of subarrays.

K = 3

0 1 2 3 (n-K) 4 5 6

n - K

[0, n-K] → n-K-0+1 → n-K+1



b - a + 1

a b

[0 n-K]

n-K-0+1 = n-K+1

[a b] ⇒ b-a+1

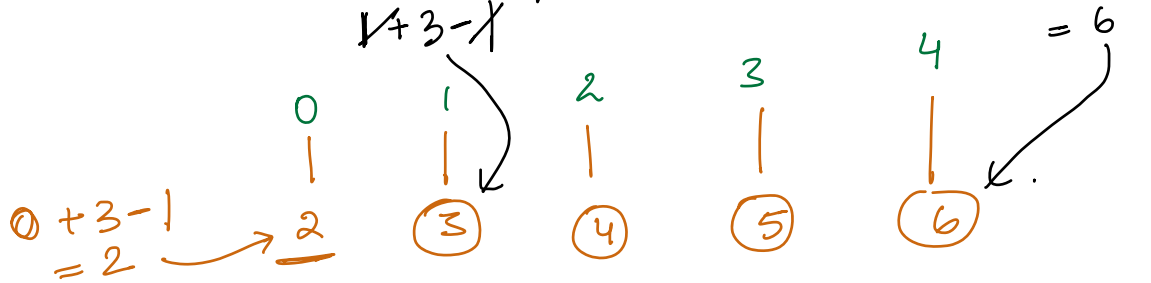
print the start & end index of all subarrays of length K

0 1 2 3 . . . n-K

- - - - - -

---

0 1 2 3 4 5 6 | K=3



ending index → i+K-1

```
for ( i = 0 ; i <= n-K ; i++) {
    print ( i  &  i+K-1 )
}
```

Q. max subarray sum with length K.

0	1	2	3	4	5	6	7	8	9
-3	4	-2	5	3	-2	8	2	-1	4

K = 5

[0, 4] → 7
[1, 5] → 8
[2, 6] → 12
[3, 7] → 16 ans
[4, 8] → 10
[5, 9] → 11

Brute Force

```
max_sum = -∞
for(i=0; i<=n-K; i++){
    j = i+K-1
    sum = 0
    for(z=i; z<=j; z++){
        sum = sum + A[z]
    }
    max_sum = max(max_sum, sum)
}
```

i: 0 → n-K → n-K+1

z: K

Exact Iterations: $((n-K+1)*(K))$

$K = [1 \quad n]$

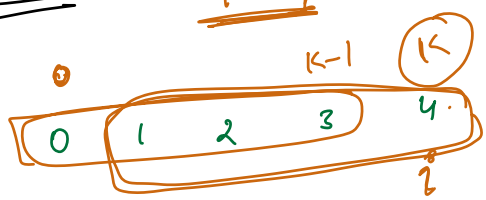
$K = n$	$K = 1$	$K \simeq n/2$
$(n-n+1)(n)$	$(n-1+1)(1)$	$(n-\frac{n}{2}+1)(\frac{n}{2})$
$= n$	$= n$	$(\frac{n}{2}+1)(\frac{n}{2})$
		n^2

TC: $O(n^2)$

Optimised

K = 4

A[0] + A[1] + A[2] + A[3]



① Calc the sum from 0 to K-1 [sum]

② Move i from K to n-1

③ Add A[i] to sum
Remove A[i-K] from sum

④ maintain the max sum

```
sum = 0
for( i = 0 ; i <= k-1 ; i++ ) {
        sum = sum + A[i]
}
max_sum = sum
for( i = K ; i < n ; i++ ) {
        sum = sum + A[i]
        sum = sum - A[i-K]
        max_sum = max (max_sum , sum)
}
return max_sum
```

K

n - K

Technique : Sliding Window.

Total Iterations = ~~k~~ + n - ~~k~~
= n

=

TC ⇒ O(n)

K=3

0	1	2	3	4	5	6
1	-3	4	2	6	9	2

max_sum

~~-∞~~

~~2~~

~~3~~

~~12~~

17

sum = 2 + 2 - 1

= 3 + 6 - (-3)

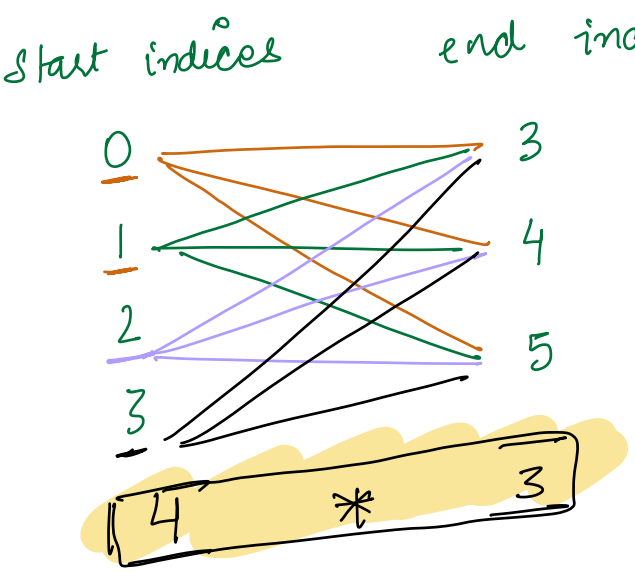
= 9 + 3 = 12

= 12 + 9 - 4 = 17

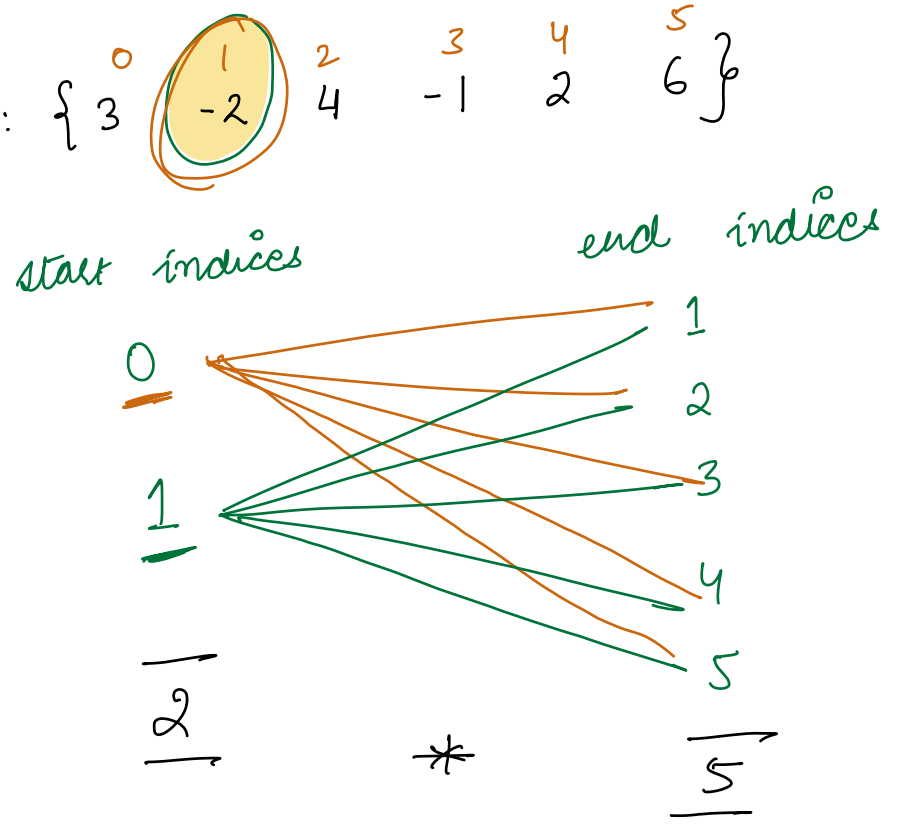
17 + 2 - 2 = 17

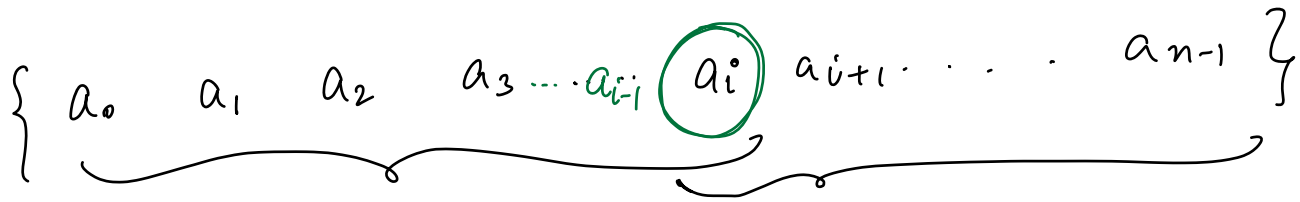
Qu: In how many subarrays index 3 is present?

arr[] : { 3 (0) -2 (1) 4 (2) -1 (3) 2 (4) 6 (5) }



arr[] : { 3 (0) -2 (1) 4 (2) -1 (3) 2 (4) 6 (5) }





$i - 0 + 1 = i + 1$

$n - i - i + i = n - i$

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

Contribution Technique

Q. Sum of all the subarray sums.

Google
Bloomberg
Amazon

{3 2 1 4}

[3] → 3 → 3

[3,2] → 5 → 3 + 2

[3,2,1] → 6 → 3 + 2 + 1

[3,2,1,4] → 10 → 3 + 2 + 1 + 4

[2] → 2 → 2

[2,1] → 3 → 2 + 1

[2,1,4] → 7 → 2 + 1 + 4

3 * 4 = 12

+

2 * 6 = 12

+

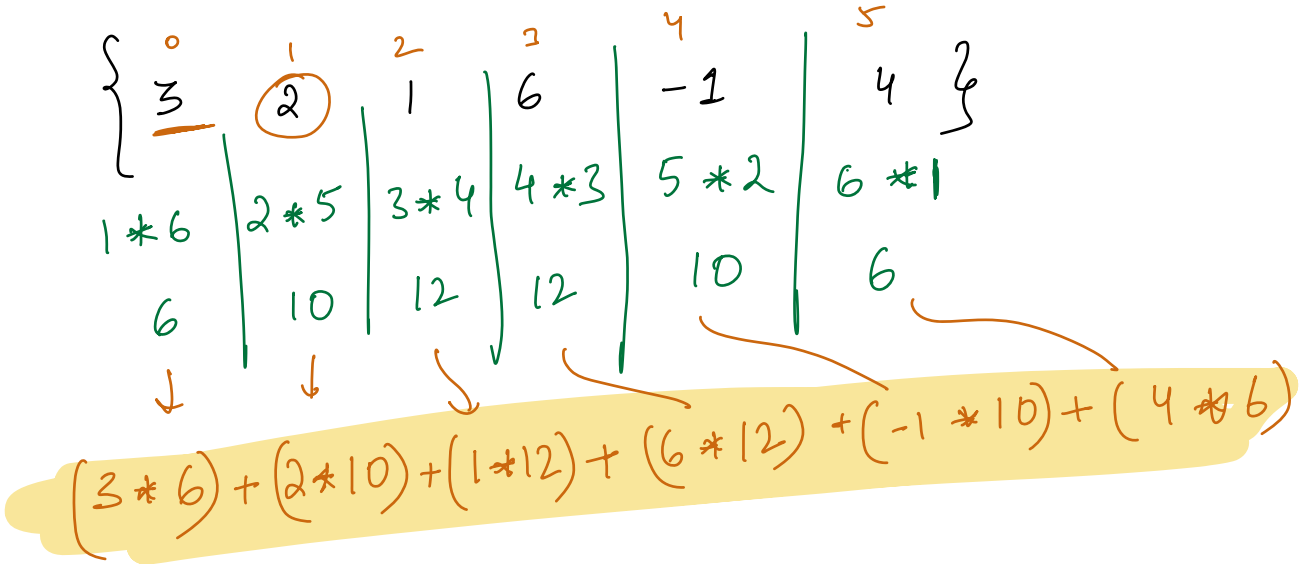
1 * 6 = 6

[1] → 1 → 1

[1, 4] → 5 → 1 + 4

[4] → 4 → 4

$(i+1) * (n-i)$ $\quad n = 6$



optimised

```
sum = 0
for (i=0; i<n; i++) {
    appears = (i+1) * (n-i)
    contribution = A[i] * appears
    sum = sum + contribution
}
```

TC

O(n)

SC

O(1)

Brute Force

```
overall_sum = 0
for( i = 0; i < n; i++ ) {
    sum = 0
    for( j = i; j < n; j++ ) {
        sum = sum + A[j]
        overall_sum += sum
    }
}
return overall_sum
```

OR

```
sum = 0
for( i = 0; i < n; i++ ) {
    for( j = i; j < n; j++ ) {
        sum = sum + A[j]
        sum += sum
    }
}
return sum
```

?

i=0

j=0 A[0] sum=

j=1 A[0] + A[1]

TC

$O(n^2)$

SC

$O(1)$