< *Question* > : Given arr[N]. Find ~~maximum~~ sum of all subarray sum.

[1] → 1

[1 , 2] → 3

[1 , 2 , 3] → 6

[2] → 2

[2 , 3] → 5

Idea -1

Idea -2 use psum[ ]

Idea -3 Carry Forward

arr [ ] → [-4 1 3 2]

0 1 2 3

[Dry - run] -

arr[ ] → [-4 1 3 2]

0 1 2 3

< Question > : Given arr[N]. Find ~~maximum~~ sum of all subarray sum.

all = {1, 2, 3}

[1] → 1

[1, 2] → 1 + 2

[1, 2, 3] → 1 + 2 + 3

[2] → 2

[2, 3] → 2 + 3

[3] → 3

sum = 20

BF Idea

⇒ Print all subarrays

instead of print, take the sum

TC: $O(N^3)$

Idea -2 use psum[ ]

using prefix sum ⇒ sum[l:r]

pf[r] − pf[l−1]

1) Create psum[] TC, SC: O(N)

2) total = 0

```
for( i : 0 → n-1) {
    for( j : i → n-1) {
        sum = pf[j] - pf[i-1]
        total += sum
    }
}
```

TC: $O(N^2)$

1, 2, 3

pf 1 3 6

i, j

0,0 ⇒ 1
0,1 3
0,2 ⇒ 6
1,1 ⇒ 2
1,2 5
2,2 ⇒ 3

} 20

0 1 2

1, 2, 3

Idea -3 Carry Forward

```
total=0
for( i: 0 → n-1) {
    cur=0
    for( j: i → n-1) {
        cur += arr[j]
        total += cur
    }
}
```

cur = 0 1+2 = 3 +3 = 6

total += 1 +3 +6

arr[l:r] + arr[r+1] = arr[l:r+1]

TC: O(N²)

arr[3:8] + arr[9] ⇒ arr[3:9]

Idea -4 arr → [3 -2 1 4]
0 1 2 3

0,0 0,1
0,1 0,2
0,2 0,3
0,3 1,1
⋮ 1,2
0,n-1 1,3

If length = n

idx 0 ⇒ n

idx 1 ⇒

i = 2
n = 6

(i+1)(n-i)

3 × 4 = 12

[3]

[3 -2]

[3 -2 1]

[3 -2 1 4]

[-2]

[-2 1]

[-2 1 4]

[1]

[1 4]

[4]

(3*4) + (-2*6) + (1* 6) + (4*4)

3 ⇒ 4

-2 ⇒ 6

1 ⇒ 6

4 ⇒ 4

- How many times an element appears in all the subarrays?

arr → [3 -2 4 -1 2 6]
0 1 2 3 4 5

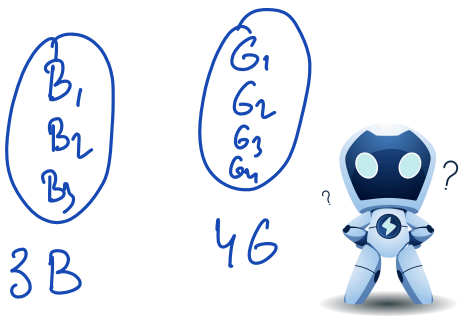
- In how many subarrays index-2 will be present?

arr → [3 -2 4 -1 2 6]
0 1 2 3 4 5

3 x 4 = 12

Generalize

B_1G_1 B_1G_2 B_1G_3 B_1G_4
B_2G_1 B_2G_2 B_2G_3 B_2G_4
B_3G_1 B_3G_2 B_3G_3 B_3G_4



' In how many subarray, ith index element will be present? '

0 1 2 3 4 5 6

start $\leq i \Rightarrow i+1$ 0, 1, 2, ... i, i+1 ... n-1

end $\geq i \Rightarrow [i, n-1] \Rightarrow$ n-i+1

$n-1-i+1 = n-i$

total subarrays = start * end = $(i+1)(n-i)$

< / > Code

```
ans = 0
for ( i : 0 → n-1 ) {
        contribution = (i+1)(n-i)
        ans += arr[i] * contribution
}
```

TC: O(N)

SC: O(1)

- **Number of subarrays of length k**

Number of subarrays with length = 1 → 6

Number of subarrays with length = 2 → 5

Number of subarrays with length = 3 → 4

Number of subarrays with length = 4 → 3

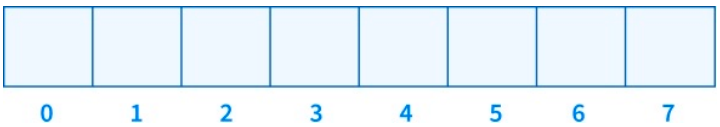
Number of subarrays with length = 5 → 2

Number of subarrays with length = 6 → 1

If len = N, how many subarrays of len = k ⇒ n − k + 1

< *Question* > : Print si and ei of every subarray of length k.

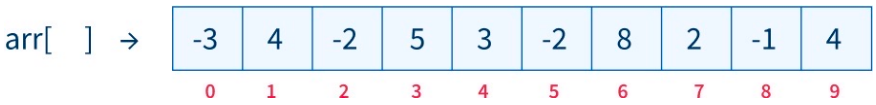
N = 8, K = 3



Si	ei
0	2
1	3
2	4
3	5
4	6
5	7

```
s = 0        e = k-1
while (e <= (n-1)) {
    print(s)
    print(e)
    s++      e++
}
```

< *Question* > : Given arr[N]. Print maximum subarray sum of subarray with length k.



K = 5

s e

0 4 ⇒ 7

1 5 ⇒ 8

2 6 ⇒ 12

3 7 ⇒ 16

4 8 ⇒ 10

5 9 ⇒ 11

ans = 16

BF Idea

For each subarray, iterate & find the sum

< / > Code

Idea -2 use psum[ ]

1) Create prefix sum

```
s = 0        e = k-1
while (e <= (n-1)) {
    sum = pf[e] - pf[s-1]
    ans = max(ans, sum)
}
```

TC: O(n)
SC: O(n)

arr[ ] →	-3	4	-2	5	3	-2	8	2	-1	4
	0	1	2	3	4	5	6	7	8	9

K = 5

s e

0 4 ⇒ ar[0]+ar[1]+ar[2]+ar[3]+ar[4]

1 5 ⇒ ar[1]+ar[2]+ar[3]+ar[4]+ar[5]

⇒ prev + ar[5] − ar[0]

2 6 ⇒ prev + ar[6] − ar[1]

3 7 ⇒ prev + ar[7] − ar[2]

4 8 ⇒ prev + ar[8] − ar[3]

5 9

From prev ans to next ans

prev + ar[e] − ar[s−1]

< / > Code

1. Create the window → Calculate sum of first K elements.

```
0, k-1
sum = 0
for (i: 0 → k-1) {
    sum += arr[i]
}
ans = sum
```

TC: O(n)

SC: O(1)

2. Consider the remaining subarrays of length K with sliding window

```
s = 1        e = k
while (e ≤ n-1) {
    sum = sum + arr[e] - arr[s-1]
    ans = max(ans, sum)
    s++      e++
}
```

0	1	2	3	4	
10	20	30	20	10	k=3

0:2 ⇒ ~~60~~ ~~70~~ 60