Ques

Given an array of integers, we need to find the sum of all possible subarrays of the array and maintain the maximum sum.

arr[] = [10, 20, 30] $n \to \frac{n(n+1)}{2}$

Subarrays :-

		sum
10	⇒	10
10, 20	⇒	30
10, 20, 30	⇒	60
20	⇒	20
20, 30	⇒	50
30	⇒	30

1) Brute force

T.C → $O(n^3)$
S.C → $O(1)$.

```
for ( s = 0; s < n; s++) {
    for (e = s; e < n; e++) {
        int sum = 0;
        for (i = s; i <= e; i++) {
            sum += arr[i];
        }
        print(sum);
    }
}
```

2) Using Prefix sum

sum(2, 2), →

```
int pf[n]    // Todo    // O(n)
                                       ] T.C -> O(n^2)
for ( s = 0; s < n; s++) {   // O(n^2)      S.C -> O(n)

    for (e = s; e < n; e++) {
        int sum = 0;
        if (s != 0) { sum = pf[e] - pf[s-1] }
        else { sum = pf[e] }
        Print (sum);
    }
}
```

idea 3 :- Carry forward

$$\begin{array}{ccc} 0 & 1 & 2 \\ 10, & 20, & 30 \end{array}$$

```
for ( s = ~~0~~ 0; s < n; s++) {        (0,0) -> 10, arr[0]

    for (e = ~~s~~ ~~0~~ ~~1~~ 2; e < n; e++) {      (0,1) -> 30, arr[0] + arr[1]
        int sum = 0;
        for (i = s; i <= e; i++) {      (0,2) -> 60, arr[0] + arr[1] + arr[2]
            sum += arr[i];
        }
        Print (sum);
    }
}
```

```
for ( s = 0; s < n ; s++) {
    int sum = 0;
  for (e = s; e < n; e++) {
        sum + = arr[e];
        print (sum);
  }
}
```

$T.C \to O(n^2)$

$S.C \to O(1)$

Console

10
30
60
20
50
30

0 1 2
[10, 20, 30]

s	e	sum
0	0	~~0~~ 10
0	1	30
0	2	60
1	1	~~0~~ 20
1	2	50
2	2	~~0~~ 30

```
int maxSum = -∞

for ( s = 0; s < n ; s++) {
    int sum = 0;
  for (e = s; e < n; e++) {
        sum + = arr[e];
        maxSum = Max(maxSum, sum);
  }
}

print (maxSum);
```

Ques) Print Sum of all subarray sums.
or add all subarrays

0 1 2
[10, 20, 30]

s	e	Subarrays	Sum
0	0	10	~~0~~ 10
0	1	10, 20	30
0	2	10, 20, 30	60
1	1	20	~~0~~ 20
1	2	20, 30	50
2	2	30	~~0~~ 30
		200	200

```
totalsum = 0;
for ( s = 0; s < n; s++) {
    int sum = 0;
    for (e = s; e < n; e++) {
        sum+ = arr[e];
        totalsum+ = sum;
    }
}
Print (totalsum);
```

T.C → $O(n^2)$
S.C → $O(1)$

idea :- Contribution technique

$$\begin{array}{ccc} 0 & 1 & 2 \\ [10, & 20, & 30] \\ \downarrow & \downarrow & \downarrow \\ 3 & 4 & 3 \end{array} \rightarrow 10\times3+20\times4+30\times3$$

s	e	subarrays	sum
0	0	10	~~0~~ 10
0	1	10, 20	30
0	2	10, 20, 30	60
1	1	20	~~0~~ 20
1	2	20, 30	50
2	2	30	~~0~~ 30
		200	200

Ques

In how many subarrays, the element at index 1 will be present?

A: [3, -2, 4, -1, 2, 6]

Ans → 10 .

In how many subarrays, the element at index 1 will be present?

A: [3, -2, 4, -1, 2, 6]

Quiz :-

In how many subarrays, the element at index 2 will be present?

A: [3, -2, 4, -1, 2, 6]

Ans → 12.

In how many subarrays, the element at index 2 will be present?

A: [3, -2, 4, -1, 2, 6]

In how many subarrays, the element at index 2 will be present?

A: [3, -2, 4, -1, 2, 6]

arr[] = { 3, -2, 4, -1, 2, 6 }

s	e
0	2
1	3
2	4
	5

⇒ 12

$$\begin{array}{l} \quad\quad\quad\ \ {\scriptstyle 0}\quad\ {\scriptstyle 1}\quad\ \ {\scriptstyle 2}\quad\ {\scriptstyle 3}\quad {\scriptstyle 4}\quad {\scriptstyle 5} \\ arr[\,] = \{\ 3,\ -2,\ 4,\ -1,\ 2,\ 6\ \} \end{array}$$

$n = 6$

$i = \underline{1}$.

s	e
0	1
1	2
	3
	4
	5

$\Rightarrow \underline{10}$.

$(1+1)(6-1)$

$\Rightarrow 2 \times 5 \Rightarrow \underline{10}$.

Ques) In how many subarrays idx i will be present?

0 .... i n-1

$$\Rightarrow \underline{(i+1)(n-i)}$$

s	e
0	i
1	i+1
2	⋮
⋮	⋮
i	n-1

$(i+1)$ choices of s; $(n-i)$ choices of e.

$(a \quad b) \rightarrow b - a + 1$

$(i \text{ to } n-1) \Rightarrow n - \cancel{1} - i + \cancel{1}$

$\Rightarrow (\underline{n-i})$

T.C -> O(n)

S.C -> O(1).

```
ans = 0;
for (i=0; i<n; i++) {
        freq = (i+1)*(n-i)
        contr = freq * arr[i];
        ans += contr;
}
return ans;
```

arr[n] = 0 1 2 . . . (n-2), (n-1)

len	start of first window	start of last window
1	0	n-1
2	0	n-2
3	0	n-3
4	0	n-4
K	0	n-K

How many subarrays of len K are there?

(0 to n-k) => n-k-0+1

=> n-k+1 =

arr[7] = 0, 1, 2, 3, 4, 5, 6   k=4

n-k+1 , k=4, n=7

↳ 7-4+1 => 4.

Ques.

Given an array of size N, print start and end indices of subarrays of length K.

$n = 8, \quad k = 3 \rightarrow \quad 8 - 3 + 1 \Rightarrow 6$

0	1	2	3	4	5	6	7
a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7

s	e
0	2
1	3
2	4
3	5
4	6
5	7

$(s, e) \Rightarrow k$

$e - s + 1 = k$

$\Rightarrow e = k + s - 1.$

$n = 8, \quad k = 3$

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

3

s	e
0	2
1	3
2	4
3	5
4	6
5	7

Ques :-

Given an array of N elements. Print maximum subarray sum for subarrays with length = K.

N = 10, k = 5

cont of subarray of len k :- n-k+1

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

10-5+1 => 6.

s	e		sum
0	4	→	7
1	5	→	8
2	6	→	12
3	7	→	16
4	8	→	10
5	9	→	11

Ans => 16.

k = 5

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

s e

e => (k-1, n-1)

$n - \cancel{x} - k + 1 + \cancel{x}$

=> n-k+1

```
ans = -∞;
s = 0
e = k-1
while (e<n) {
    sum = 0;
    for (i = s; i <= e; i++) {
        sum += arr[i];
    }
    ans = Max (ans, sum);
    s++;
    e++;
}
Print (ans);
```

$T.C \rightarrow O(n-k+1) \times k$, $S.C \rightarrow O(1)$.

when $k = 1$

$(n-1+1) \times 1$

$\downarrow$

$O(n)$

$k = n$

$(n-n+1) \times n$

$\downarrow$

$O(n)$

$k = n/2$

$\left(n - \frac{n}{2} + 1\right) * \frac{n}{2}$

$\left(\frac{n}{2} + 1\right) * \frac{n}{2}$

$\frac{n^2}{4} + \frac{n}{2}$

$\Rightarrow O(n^2)$

idea 2 :- using Prefix sum

```
int pf[n]   // Todo
ans = -∞ ;

s = 0
e = k-1
while ( e<n ) {
        sum = 0;
        if (s == 0) { sum = pf[e] }
        else {  sum = pf[e] - pf[s-1] }

        ans = Max (ans, sum);

        s++;
        e++;
}
Print (ans);
```

$T.C \rightarrow O(n-k+1)$ when $k=1$;

$O(n)$

$S.C \rightarrow O(n)$

ideas :- Sliding window :-

N = 10, k = 5

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

s	e	sum
0	4	7
1	5	7 + arr[5] - arr[0] = 7 + (-2) - (-3) = 8
2	6	8 + arr[6] - arr[1] = 8 + 8 - 4 => 12
3	7	12 + arr[7] - arr[2] = 12 + 2 - (-2) => 16
4	8	16 + arr[8] - arr[3] = 16 + (-1) - 5 => 10
5	9	10 + arr[9] - arr[4] = 10 + 4 - 3 => 11

sum of previous window = sum (s-1, e-1)

↓

sum of current window (s e)

sum + arr[e] - arr[s-1];

s	e	sum
0	4	7
1	5	7 + arr[5] - arr[0] = 7 + (-2) - (-3) = 8
2	6	8 + arr[6] - arr[1] = 8 + 8 - 4 => 12
3	7	12 + arr[7] - arr[2] = 12 + 2 - (-2) => 16
4	8	16 + arr[8] - arr[3] = 16 + (-1) - 5 => 10
5	9	10 + arr[9] - arr[4] = 10 + 4 - 3 => 11

```
ans = -∞;

// first window
s = 0, e = k-1
sum = 0;
for (i= s; i <= e; i++) {          → k times
      sum += arr[i];
}
        → ans = sum;
print (sum);        //sum (0 to k-1)

s++;          ( 1 to k )
e++;

while ( e < n ) {        // n - k+1 times

      sum = sum + arr[e] - arr[s-1];
      print (sum); // ans = Max(ans, sum);

      s++;
      e++;
}
```

T.C → n-k + k+1 => O(n)

S.C → O(1)

0 1 2 3 4

arr[] = [1, 4, 5, 2, 4]

0 1 2 3 4 5

2, 3, -1, 4, 2, 1 B = 4.

L	R
4	0
3	1
2	2
1	3
0	4