Todays Content:

a. Sum of all Subarray Sums

b. Man Subarray sum of len = k

c. Min Swaps requried to bring all ele $\le k$ together.

<u>Note</u>: If missed last class

a) Go watch recording ——

Printing all Sub Array Sums:

Ex: $ar[3] = \{ \overset{0}{3} \quad \overset{1}{4} \quad \overset{2}{2} \}$

Sub Arrays:

		sums
[0 0]	{3} →	3
[0 1]	{3 4} →	7
[0 2]	{3 4 2} →	9
[1 1]	{4} →	4
[1 2]	{4 2} →	6
[2 2]	{2} →	2

```
void printSum (int ar[]){
    int N = ar.length;
    for( int s=0; s< N; s++){
        int sum=0;
        for(int e=s; e< N; e++) {
            sum = sum+ar[e]
            //subarray sum [s..e]
            println(sum)
        }
    }
}
```

Q) Given an ar[N] return sum of all SubArray Sum

Ex: $ar[3] = \{ \overset{0}{3} \quad \overset{1}{4} \quad \overset{2}{2} \}$

Sub Arrays:

		sums
[0 0]	{3} →	3 +
[0 1]	{3 4} →	7 +
[0 2]	{3 4 2} →	9 +
[1 1]	{4} →	4 +
[1 2]	{4 2} →	6 +
[2 2]	{2} →	2

Final ans = 31

```
long SubSums (int ar[]){        TC: O(N^2)
                                SC: O(1)
    long total = 0;
    int N = ar.length;
    for( int s=0; s< N; s++){
        int sum=0;
        for(int e=s; e< N; e++) {
            sum = sum+ar[e]
            //subarray sum [s..e]
            total = total + sum
        }
    }
    return total;
}
```

Optimization Idea:

In Question if we see words like Sum of all

Technique: Contribution Technique: Add contribution of individual arr[] element in final ans.

Ex: $ar[3] = \begin{array}{ccc} 0 & 1 & 2 \\ \{3 & 4 & 2\} \end{array}$

SubArrays:

[0 0]	{3}
[0 1]	{3 4}
[0 2]	{3 4 2}
[1 1]	{4}
[1 2]	{4 2}
[2 2]	{2}

Contribution:

ele	occurence	contribution
3	3	$3 * 3 = 9$
4	4	$4 * 4 = 16$
2	3	$2 * 3 = 6$

Sum of all Contributions = 31

Ex: $ar[4] = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ \hline 2 & 8 & -1 & 4 \end{array}$

Sub Arrays:		Sub Array Sums
[0 0]	{2}	2
[0 1]	{2 8}	10
[0 2]	{2 8 -1}	9
[0 3]	{2 8 -1 4}	13
[1 1]	{8}	8
[1 2]	{8 -1}	7
[1 3]	{8 -1 4}	11
[2 2]	{-1}	-1
[2 3]	{-1 4}	3
[3 3]	{4}	4

Contribution:

ele		occurence	contribution
2	*	4	8
8	*	6	48
-1	*	6	-6
4	*	4	16

Sum of all Contributions = 66

Q) In how many subarray a particular index will be present?

Ex1:

	0	1	2	3	4	5
ar[6] =	{ 3	-2	4	-1	2	6 }
start:	✓	✓	✓	✓	*	*
end :	x	x	x	✓	✓	✓

In how many subarrays index 3 is present?

start ind	end ind
0	3
1	4
2	5
3	

Total Subarrays = 4 * 3 = 12

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

	0	1	2	3	4	5	
ar[6] =	{ 3	-2	4	-1	2	6 }	
start =	✓	✓	*	*	*	*	= 2 points
end =	*	✓	✓	✓	✓	✓	= 5points

In how many subarrays index 1 is present?

start	end
0	1
1	2
	3
	4
	5

Total Subarrays = 2 * 5 = 10

Final obs: In ar[N] in how many subarrays index i present.

	0	1	2		i-1	i	i+1	i+2		n-2	n-1	
ar[N] = {	a_0	a_1	a_2	.	a_{i-1}	a_i	a_{i+1}	a_{i+2}	....	a_{n-2}	a_{n-1}	}
start =	✓	✓	✓	...	✓	✓	*	*	..	*	*	
end =	*	*	*	...	*	✓	✓	✓	..	✓	✓	

start = i+1

a b b-a+1
[0 i] i-0+1 = i+1

end = N-i

a b b-a+1
[i, N-1] N-~~1~~-i+~~1~~

Subarrays i^th index
$(i+1)*(N-i)$

Final Conclusion: Count of Subarrays with i^th index = $(i+1)*(N-i)$

Dry Run:

N=4 Ex: ar[4] =

	0	1	2	3
	2	8	-1	4
	(i+1)(N-i)	(i+1)(N-i)	(i+1)(N-i)	(i+1)(N-i)
#Count of Subarrays with given index	= (0+1)(4-0)	(1+1)(4-1)	(2+1)(4-2)	(3+1)(4-3)
	= 4	6	6	4
#Contribution	= 8 +	48 +	-6 +	16

```
long SubSums(int ar[]){   TC:O(N) SC:O(1)
    int N= ar.length;
    long total = 0;
    for(int i = 0; i < N; i++){
        // In how many sub ar[i] present
        long freq = (i+1)(N-i)
        long con = freq*ar[i]
        total = total + con
    }
    return total;
}
```

2Q) Given ar[N] elements, return Max Subarray Sums of len = k

Constraints:

$1 <= N <= 10^5$
$1 <= k <= N$
$-10^6 <= ar[i] <= 10^6$

$N = 10^5$, Each ele $= 10^6$, max sum $= 10^{11}$

Ex:

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

Subarrays

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

Max Subarray Sum = 16

ans = 16

Idea: For every subarray of len = k, iterate & get sum & get overall Max.

Q: No: of Subarrays of len = k in ar[N] = N-k+1

TC: (Total subarrays) * (TC for Each)

TC: (N-k+1) * (k)

✓ $k=N$: $(N-N+1)*(N) = N$

✓ $k=1$: $(N-1+1)*(1) = N$

✓ $k=N/2$: $(N-N/2+1)*(N/2)$

$= (N/2+1)(N/2) \approx \frac{N^2}{4}$

$= O(N^2)$

```
long maxSub(int ar[], int k){
    int N = ar.length
    int s = 0, e = k-1;
    long ans = INT_MIN
    while(e < N){
        long sum = 0; // Sub : [s ... e]
        for(int i = s; i <= e; i++){
            sum = sum+ar[i]
        }
        s = s+1; e = e+1;
        if(ans < sum) {ans = sum}
    }
    return ans;
}
```

TC: O(N^2)
SC: O(1)

Idea2:

```
long maxSub(int ar[], int N, int k){
    int N= ar.length
    long psum[N]; // Construct  →  TC:O(N)
    int s = 0, e = k-1,                    +
    long ans = INT_MIN  .
    while(e < N){  →  TC:O(N-K+1)
        long sum=0; // Sub:[s...e]
        if(s==0){ sum = Psum[e] }  //sum:[0..e]
        else{
            sum = Psum[e] - Psum[s-1]
        }
        if(ans<sum){ans = sum}
        s=s+1; e=e+1;
    }
    return ans;
}
```

TC:O(N+N-K)

TC:O(2N) = O(N)

SC:O(N)

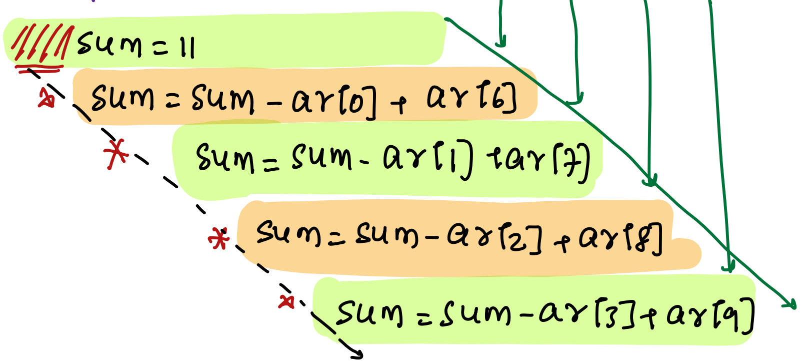
Optimizatim Idea:

In Question if subarray size fixed

Technique name: Sliding Window + Data we Slide

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

k=6



sum = 11

sum = sum − ar[0] + ar[6]

sum = sum − ar[1] + ar[7]

sum = sum − ar[2] + ar[8]

sum = sum − ar[3] + ar[9]

s e

0 5 sum = 11 : Iterate & Calculate

			Delete	Add	value
1	6	sum = sum −	ar[0] +	ar[6] =	11 − 3 + 8 = 16
2	7	sum = sum −	ar[1] +	ar[7] =	16 − 4 + 2 = 14
3	8	sum = sum −	ar[2] +	ar[8] =	14 − (−2) + 1 = 17
4	9	sum = sum −	ar[3] +	ar[9] =	17 − 5 + 4 = 16

ans = 17

```
long  sub(int arr[], int k){
    long ans = INT_MIN;
    int N= arr.length;
    long sum = 0;
    for( int i=0; i<k; i++){  →  k iterations
        sum = sum+arr[i]
    }
    if (ans < sum){ ans = sum} // 1st subarray

    int s=1,  e = k;
    while(e<n){  →  N-k+1
        //get subarr sum [s..e] using sliding window
        sum = sum - arr[s-1] + arr[e]
        if (ans < sum){ ans = sum}
        s++; e++;
    }
    return ans;
}
```

Total TC: O(N)

SC: O(1)

38) Min Swaps Required to bring all $\underline{\underline{\text{ele} \leq B}}$ together

$$B=10 \quad ar[10] = \{\ \overset{0}{14}\quad \overset{1}{2}\quad \overset{2}{9}\quad \overset{3}{21}\quad \overset{4}{24}\quad \overset{5}{8}\quad \overset{6}{30}\quad \overset{7}{19}\quad \overset{8}{5}\quad \overset{9}{10}\ \}$$