Todays Content

1. Subarrays Introductin
2. Print all subarrays from s..e
3. Print all subarray sums
    - 3a Optimizatin using Prefin Sum
    - 3b Optimizatin using Contributin Technique

Introduction to Subarrays:

Definition:

Subarray is continous part of an array

1 element or complete array is also considered as subarray.

#Note: Subarray considered left to right

	0	1	2	3	4	5	6	7	8
ar[9] = {	4	1	2	3	-1	6	9	8	12 }

{ 2 3 -1 6 } : Yes

{ 9 } : Yes

{ 4 1 2 3 -1 6 9 8 12 } : Yes

{ 4 2 } No

{ 1 2 6 } No

Representation of Subarray:

With start & end index of subarray

	0	1	2	3	4	5	6	7	8
ar[ ] = {	4	1	2	3	-1	6	9	8	12]

# start =	end =	
3	8	{ 3 -1 6 9 8 12 }
2	5	{ 2 3 -1 6 }

How many subarrays start from index:0

	0	1	2	3	4
Eg1: ar[ ] = {	4	2	3	7	8 }

#start:

[0 0] = {4}

[0 1] : {4 2}

[0 2] : {4 2 3}

[0 3] : {4 2 3 7}

[0 4] : {4 2 3 7 8}

#start:1

[1 1] : {2}

[1 2] : {2 3}

[1 3] : {2 3 7}

[1 4] : {2 3 7 8}

#Total Subarrays:

	0	1	2	3	4
ar[ ] = {	4	2	3	7	8 }

→ → → → } #Sub = 5

#Total subarrays len = 5

5+4+3+2+1 = 15

	0	1	2	3	4
ar[ ] = {	4	2	3	7	8 }

→ → → } #Sub = 4

#Total subarrays len = N

N + N-1 + N-2 + .. 1 = $\frac{(N)(N+1)}{2}$

	0	1	2	3	4
ar[ ] = {	4	2	3	7	8 }

→ → } #Sub = 3

	0	1	2	3	4
ar[ ] = {	4	2	3	7	8 }

→ → } #Sub = 2

	0	1	2	3	4
ar[ ] = {	4	2	3	7	8 }

→ #Sub = 1

Q0: Print a given subarray

#start #end

```
void printSub(vector<int> &arr, int i, int j){ TC: O(N) SC: O(1)
    for(int k=i; k<=j; k++){
        print(arr[k]+" ");
    }
}
```

Q1: Print all subarrays.

Ex: arr[4] = { 3 7 2 9 }
(0 1 2 3)

#Subarrays	Output
{0 0}	: {3}
{0 1}	: {3 7}
{0 2}	: {3 7 2}
{0 3}	: {3 7 2 9}
{1 1}	: {7}
{1 2}	: {7 2}
{1 3}	: {7 2 9}
{2 2}	: {2}
{2 3}	: {2 9}
{3 3}	: {9}

start ↑ end

// arr[N] = { 0 1 2 .. [i] [i+1] [i+2] .. [N-1] }

```
void printAll(vector<int> &arr){
    int N = arr.size();
    for(int i=0; i<N; i++){ i: start
        for(int j=i; j<N; j++){ j: end
            //subarray [i..j]
            for(int k=i; k<=j; k++){
                print(arr[k]);
            }
            print("\n");
        }
    }
}
```

TC: # O(N^2) * N = O(N^3)

Calculate & Return sum of all subarray sums.

Eg: ar[3] = { 3 4 2 }
(0 1 2)

Subarrays:

		#Sum
[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

Total = 31

#Ideal: Generate all subarray sums & add in total & return it.

```
long printAll(vector<int> &ar){
    long tsum = 0;
    int N = ar.size();
    for(int i = 0; i < N; i++){ i: start
        for(int j = i; j < N; j++){ j: end
            //subarray [i..j]
            long sum = 0; // 1 subarray sum
            for(int k = i; k <= j; k++){
                sum = sum + ar[k]
            }
            tsum = tsum + sum;
        }
    }
    return tsum;
```

TC: # $O(N^2)$ * $O(N)$ = $O(N^3)$ SC: $O(1)$

N^2 subarrays

Iterate & get sum for 1 subarray

#idea2 Generate all subarray sums using pfsum & add in total.

```
long printAll(vector<int> &ar){ TC: O(N + N^2/2) = O(N^2)   SC: O(N)
    int N = ar.size();

    long pf[N], sum = 0;                    // generate pf
    for(int i=0; i<N; i++){
        sum = sum + ar[i];
        pf[i] = sum;
    }

    long tsum = 0;                          // generate all subarray sums
    for(int i = 0; i<N; i++){ i: start
        for(int j= i; j<N; j++){ j: end
            //subarray [i..j] using pf[] = pf[j] - pf[i-1];
            if(i==0){ //[0..j]
                tsum += pf[j]
            }else{
                tsum += pf[j] - pf[i-1];
            }
        }
    }

    return tsum;
}
```

pf[] : O(N)

#ideas: In Question, Calculate sum of all we apply Contribution Technique

Contribution Technique = Add contribution of each ele in final ans.

Eg: ar[3] = { 3 4 2 } (0 1 2)

Subarrays:

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

Contribution:

ele	occurence	contribution
3	* 3	= 9
4	* 4	= 16
2	* 3	= 6
Total Sum		= 31

Eg: ar[4] = { 2 8 -1 4 } (0 1 2 3)

Subarrays:

[0 0]	{2}
[0 1]	{2 8}
[0 2]	{2 8 -1}
[0 3]	{2 8 -1 4}
[1 1]	{8}
[1 2]	{8 -1}
[1 3]	{8 -1 4}
[2 2]	{-1}
[2 3]	{-1 4}
[3 3]	{4}

Contribution:

ele	occurence	contribution
2	* 4	= 8
8	* 6	= 48
-1	* 6	= -6
4	* 4	= 16
Total Sum		= 66

Catch:

To calculate contribution of element we need to get it's occurences.

Ocurences: In this problem, it is no: of times ar[i] comes in subarray.

In how many subarrays a particular index i will be present

	0	1	2	3	4	5
Eg: ar[6] = {	3	-2	4	-1	2	6
#start =	✓	✓	✗	✗	✗	✗
#end =	✗	✓	✓	✓	✓	✓

#Subarrays : {s..e}

In how many subarrays index 1 is present

startInd	endInd	Total Subarrays
0	1	2 * 5 = 10.
1	2	
	3	
	4	
	5	

In how many subarrays index 2 is present

	0	1	2	3	4	5
Eg: ar[6] = {	3	-2	4	-1	2	6 }
#start =	✓	✓	✓	✗	✗	✗
#end =	✗	✗	✓	✓	✓	✓

startInd	endInd	Total Subarrays
0	2	3 * 4 = 12
1	3	
2	4	
	5	

In how many subarrays index i is present

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} }

a b b-a+1

#start = { ✓ ✓ ✓ .. ✓ ✓ ✗ ✗ .. ✗ ✗ } #[0..i] = i+1

#end = { ✗ ✗ ✗ ... ✗ ✓ ✓ ✓ . ✓ ✓ } #[i..N-1] = N-i

#subarrays = start * end = (i+1)(N-i)

#Tracing:

i	0	1	2	3
N = 4 ar[4] = {	2	8	-1	4 }
#(i+1) =	1	2	3	4
#(N-i) =	4	3	2	1
#occurency =	4	6	6	4

#Contribuh = 8 + 48 - 6 + 16 = 66

```
long subSum(vector<int> &ar){ TC:O(N) SC:O(1)
    int N=ar.size();
    long sum=0;
    for(int i=0; i<N; i++){
        long occ = (i+1)*(N-i)
    } sum = sum + ar[i]*occ;
    return sum;

}
```