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.

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

{ 3 -1 6 9 8 } subarray

{ 2 3 -1 6 } subarray

{ 9 } subarray

{ 4 1 2 3 -1 6 9 8 12 } subarray

{ 4 2 } Not

{ 1 2 6 } Not

Representation of subarray:

start & end index

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

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

How many subarrays start from ?

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

#start: 0

[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:

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

#Sub = 5

#Total subarrays len = 5

Sub = 5 + 4 + 3 + 2 + 1 = $\frac{(5)(5+1)}{2}$ = 15

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

#Sub = 4

#Total subarrays len = N

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

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

#Sub = 3

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

#Sub = 2

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

#Sub = 1

Q0: Print a given Subarray

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

Q1: Print all Subarrays.

Eg: ar[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}

// ar[N] = { 0 1 2 .. [i] i+1 ..... N-1 }
(i: start)

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

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

Calculate & Return sum of all subarrays sums.

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

Subarrays:

		#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

#ans = 31

Eg2:
(0 1 2 3)
ar[4] = { 2 8 -1 4 } #ans = 66

#Ideal: For every subarray iterate & cal sum, add in total.

```
long printAll(vector<int> &ar){
    long total=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;
            for(int k=i; k<=j; k++){
                sum=sum+ar[k];
            }
            total+=sum;
        }
    }
    return total;
}
```

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

#idea2 : Create pfSum[] & use it to get sum for every subarray.

↳ Prefix ↳ Sum using pf[]

```
long printAll(vector<int> &ar){ T:O(N + N^2 * O(1)) = O(N^2)  SC: O(N)
                                  ↳ #subarrays
    int N = ar.size();
    long pf[N]; // pf[i] = sum of all elements from [0..i]
    long sum = 0;
    for(int i=0; i<N; i++){
        sum = sum + ar[i]; # 1. carry forward
        pf[i] = sum;       # 2. Store in array
    }

    long total = 0;
    for(int i=0; i<N; i++){
        for(int j=i; j<N; j++){
            # Subarray [i..j] = Pf[j] - Pf[i-1]
            long sum = 0;
            if(i==0){ # 0..j
                sum = Pf[j]
            }else{
                sum = Pf[j] - Pf[i-1];
            }
            total = total + sum;
        }
    }
}
```

Ideas: In Question: Sum of all → Contribution Technique.

Contribution Technique: Add contribution of each individual element

Eg: arr[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: arr[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 occurence of element

Occurences: In how many subarrays element arr[i] occurs.

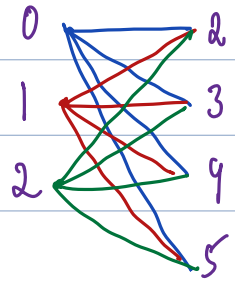
In how many subarrays a particular index 2 will be present

	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

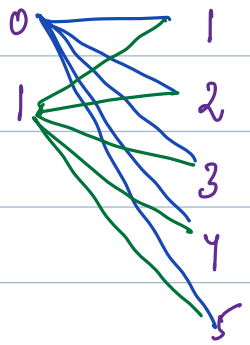


#s * #e = 3 * 4 = 12.

In how many subarrays index 1 is present

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

startInd endInd Total Subarrays



#s * #e = 10

In ar[N] 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} }	
#start : {	✓	✓	✓ ..	✓	✓	*	*	*	* }	[0..i] = i+1
#end : {	*	*	* ..	*	✓	✓	✓	✓	✓ }	[i..N-1] = N-i

#subarrays in which ar[i] is present = (i+1)(N-i)

#Tracing:

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

#(i+1) =

#(N-i) =

#subarrays

```
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++){
        #add contribution of ar[i]
        long ocu = (i+1)(N-i)
        sum = sum + ar[i]*ocu
    }
    return sum;
}
```