SUBARRAY

A very good Evening to Eve one ☺

Today's Content

01. Subarray basics
02. Printing all subarrays
03. Printing all subarrays sums
04. Return total sum of all subarrays sums

↳ Google

Subarray

→ Continous part of an array is called a subarray

01. Single element is also a subarray
02. Whole array is also a subarray
03. Subarray is considered from left to right

ar[9] =

4	1	2	3	-1	6	9	8	12
0	1	2	3	4	5	6	7	8

Check if below subarrays are correct or not

Subarrays		Indices
{2, 3, -1, 6}	→ Yes	[2 5]
{2 9}	→ No	
{10}	→ No	
{9}	→ Yes	
{8, 12}	→ Yes	

Properties of subarray

01. Given st index & end index of subarr

$$len = end - st + 1$$

02. arr = { 4, 2, 10, 3, 12, -2, 5 }
0 1 2 3 4 5 6

subarrays starting with index 1

(1, 1) = {2}
(1, 2) = {2, 10}
(1, 3) = {2, 10, 3}
(1, 4) = {2, 10, 3, 12}
(1, 5) = {2, 10, 3, 12, -2}
(1, 6) = {2, 10, 3, 12, -2, 5}

Ans # 6

03. arr = { 4, 2, 10, 3, 12, -2, 5 }
0 1 2 3 4 5 6

subarr starting from index 3

(3, 3) = {3}
(3, 4) = {3, 12}
(3, 5) = {3, 12, -2}
(3, 6) = {3, 12, -2, 5}

Ans # 4

04 arr = { 4, 2, 10, 3 }
0 1 2 3

st=0	st=1	st=2	st=3
(0,0)	~~(1,0)~~	(2,2)	(3,3)
(0,1)	(1,1)	(2,3)	1
(0,2)	(1,2)	2	
(0,3)	(1,3)		
4	3		

Total no. of subarrays = 4+3+2+1
= 10

Total no. of subarrays of (array of size N)

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

01. Given an arr[N], s, e, print subarr from [s e]

s → st point

e → end point

arr = { 4, 2, 10, 3, 12, -2, 5 } s=3, e=6

0 1 2 3 4 5 6

arr = { 4, 2, 10, 3, 12, -2, 5 } s=2, e=5

0 1 2 3 4 5 6

Idea → Iterate from s to e & print each & every element

```
void print (arr[], s, e)
{
    for (int i=s ; i<=e ; i++)
    {
        print (arr[i]);
    }
}
```

TC = O(n)

SC = O(1)

02. Given ar[N], print all subarrays

Note :- Print each subarr in new line

Constraints :- $1 \le N \le 10^2$

$1 \le A[i] \le 10^5$

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

st=0 → (0,0) = {2}

(0, 1) = {2, 8}

(0, 2) = {2, 8, -1}

(0, 3) = {2, 8, -1, 4}

st=1

(1, 1) = {8}

(1, 2) = {8, -1}

(1, 3) = {8, -1, 4}

st=2

(2, 2) = {-1}

(2, 3) = {-1, 4}

(3, 3) = {4}

Idea → To print a subarr, we need the st & end point

i j	i j	i j	i j
(0,0)	(1,1)	(2,2)	(3,3)
(0,1)	(1,2)	(2,3)	
(0,2)	(1,3)		
(0,3)			

i → 0 to n-1

j → i to n-1

```
void printallsubarr(arr[])
{
    for(i=0; i<n; i++){ // i → st pt
        for(j=i; j<n; j++){ // j → end pt
            st=i, end=j
            for(k=st; k<=end; k++){
                print(arr[k]);
            }
            println();
        }
    }
}
```

TC = O(n³)

SC = O(1)

Best TC

Total no. of subarrays = $\frac{n(n+1)}{2}$

TC to print 1 subarray = $O(n)$

Total TC = $\frac{n(n+1)}{2} * n$

$\simeq n^2 * n$

$\simeq O(n^3)$

02. Given N arr elements, print each subarr sum

Note :- Print each subarr sum in new line

Constraints :- $1 \le N \le 10^3$

$1 \le arr[i] \le 10^5$

arr [4] =

6	8	-1	7
0	1	2	3

subarr sum

[0 0] 6

[0 1] 14

[0 2] 13

[0 3] 20

[1 1] 8

[1 2] ⟶ 7

[1 3] ⟶ 14

[2 2] -1

[2 3] 6

[3 3] ⟶ 7

Idea → For every subarray, calculate sum & print it.

```
void sumsubar (arr[]){
    for (i=0 ; i<n ; i++){ // i → st pt
        for (j=i ; j<n; j++){ // j → end pt
            st=i , end=j , sum=0
            for(k=st ; k≤end ; k++){
                sum += arr[k]
            }
            println(sum);
        }
    }
}
```

TC : $O(n^3)$

SC : $O(1)$

02. Use prefix sum (since calculating the sum for a given range)

int [] psum = new int [n]; ⟶ {TODO}

```
void prefsumsub ( int [] ar )
    0. Construct psum array → {TODO}
    for ( i=0 ; i<n ; i++){ // i → st pt
        for ( j=i ; j<n; j++){ // j → end pt
            if (i==0) println ( psum[j]);
            else println ( psum[j] - psum[i-1]);
        }
    }
}
```

TC : $O(n^2)$

SC : $O(n)$

SC : $O(1)$

03. Use carry forward approach

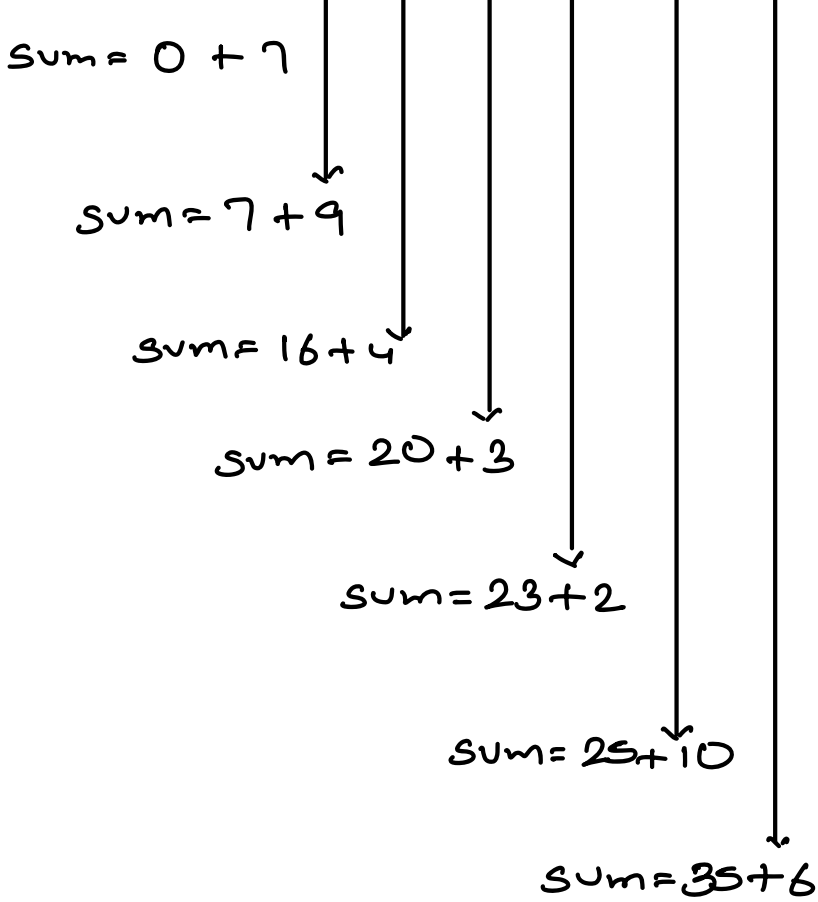
Q → Given an arr[N], print all subarrays sums starting from index 3

	0	1	2	3	4	5	6	7	8	9
ar[10] =	3	8	4	7	9	4	3	2	10	6

sum = 0

subarr sum

(3, 3)	7
(3, 4)	16
(3, 5)	20
(3, 6)	23
(3, 7)	25
(3, 8)	35
(3, 9)	41



index = 3

```
void printsum ( int[] arr, int k)
{
    int sum = 0
    for ( j = k ; j < n ; j++)
    {
        sum = sum + ar[j]
        print (sum):
    }
}
```

TC = O(n)

SC = O(1)

K=3 → print all subarr sum starting from 3

K=4 → print all subarr sum starting from 4

K=5 → print all subarr sum starting from 5

K=i → print all subarr sum starting from i

* Back to original question

```
for (int i=0; i<n; i++) {
    int sum = 0
    for ( j=i ; j<n; j++ )
        sum = sum + ar[j]
        print (sum):
    }
}
```

TC = $O(n^2)$

SC = $O(1)$

10:21 → 10:31 pm

04 Given an arr[n], return sum of all subarr sum.

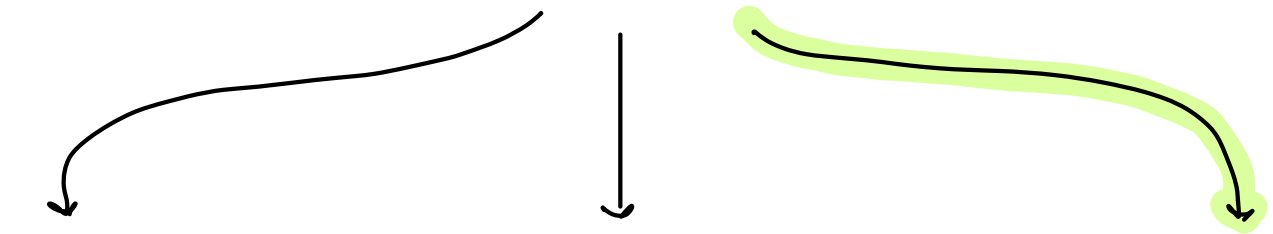
arr[4] = { 6, 8, -1, 7 } ⟶ 94

0 1 2 3

subarr	sum
[0 0]	6
[0 1]	14
[0 2]	13
[0 3]	20
[1 1]	8
[1 2]	7
[1 3]	14
[2 2]	-1
[2 3]	6
[3 3]	7

Return sum of all subarrays sums

Brute force → For every subarray, get the sum & add it to the total sum



Approach 1

3 nested loops

→ TC : $O(n^3)$

→ SC : $O(1)$

Approach 2

Prefix sum

TC : $O(n^2)$

SC : $O(n)$

Approach 3

Carry forward

TC : $O(n^2)$

SC : $O(1)$

```
totsum = 0
for (int i=0; i<n; i++) {
    int sum = 0
    for ( j=i ; j<n ; j++ )
        sum = sum + ar[j]
        totsum = totsum + sum;
    }
}
```

TC : $O(n^2)$

SC : $O(1)$

TODO → { Don't take total sum, instead use the sum outside for the loop }

| 2 | 3 | -1 | → Wrong ans

Google → Constraint → TC: O(n)
SC: O(1)

Find Idea → {Contribution technique}

arr = { 3 -2 4 -1 2 6 }
0 1 2 3 4 5

In how many subarrays, index 1 will be present ?

st	end
0	1
1	2
	3
	4
	5

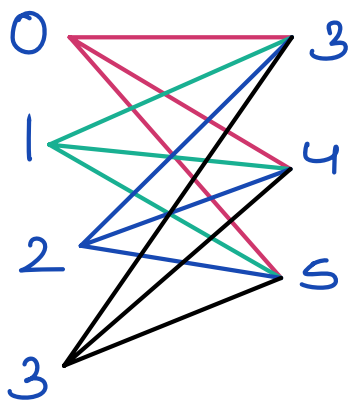
Total no. of subarr = 2 * 5 = 10

arr = { 3 -2 4 -1 2 6 }

0 1 2 3 4 5

In how many subarrays, index 3 will be present ?

st end



(0,3) (1,3) (2,3) (3,3)

(0,4) (1,4) (2,4) (3,4)

(0,5) (1,5) (2,5) (3,5)

Total no. of subarrays

4 * 3 = 12

* Given ar[N], how many subarr will contain the index i ?

arr = { a_0 a_1 a_2 .... a_{i-1} a_i a_{i+1} .... a_{n-2} a_{n-1} }

st [0 - i]

↓ (i - 0 + 1)

(i + 1)

end [i n-1]

↓ (n-1 - i + 1)

(n - i)

Total no. of subarray containing index i = (i+1) * (n-i)

arr[4] = {6, 8, -1, 7}

	0	1	2	3
(i+1) =	1	2	3	4
	*	*	*	*
(n-i) =	4	3	2	1

Total no. of subarr = 4 6 6 4

4 → In 4 subarr, 7 will be present

4 is the no. of times 6 is going to contribute

6 is the no. of times 8 is going to contribute

In 6 subarr -1, will be present

arr[4] = {6, 8, -1, 7}

0 1 2 3

subarr	sum	
[0 0]	6	{6} = 6
[0 1]	14	{6, 8} = 6 + 8
[0 2]	13	{6, 8, -1} = 6 + 8 + (-1)
[0 3]	20	{6, 8, -1, 7} = 6 + 8 + (-1) + 7
[1 1]	8	{8}
[1 2]	7	{8, -1}
[1 3]	14	{8, -1, 7}
[2 2]	-1	{-1}
[2 3]	6	{-1, 7}
[3 3]	7	{7}

Total contribution of each & every ele

no. of times an ele is contributing

= 6 * 4
+ 8 * 6
+ -1 * 6
+ 7 * 4

94

```
int  totalsum = 0

for ( i=0 ; i<n ; i++ ){
    int  nt = (i+1) * (n-i);

    totsum += nt * ar[i];
}
```

TC = O(n)

SC = O(1)

Contribution technique → Participation of each & every element

```
for (i=0; i<n; i++) {
    → int sum = i;
    {=
}
3
```

i = 0

i = 1