

Todays Content

1. Subarrays Introduction
2. Print all subarrays from S..e
3. Print all Subarray sums
 - 3a Optimization using Prefix Sum
 - 3b Optimization using Contribution Technique

Introduction to Subarrays:

Definition:

Subarray is continuous part of an array

1 element or complete array is also considered as subarray.

Note: Subarray considered left to right

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

{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

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

Start = end =

3 8 {3 -1 6 9 8 12}

2 5 {2 3 -1 6}

how many subarrays start from index: 0

$$ar[] = \{ \underset{4}{\downarrow} \underset{2}{\downarrow} \underset{3}{\downarrow} \underset{7}{\downarrow} \underset{8}{\downarrow} \}$$

#start:

$$[0 \ 0] = \{ 4 \}$$

#start: 1

$$[0 \ 1], \{ 4 \ 2 \}$$

$$[1 \ 1], \{ 2 \}$$

$$[0 \ 2]: \{ 4 \ 2 \ 3 \}$$

$$[1 \ 2]: \{ 2 \ 3 \}$$

$$[0 \ 3]: \{ 4 \ 2 \ 3 \ 7 \}$$

$$[1 \ 3]: \{ 2 \ 3 \ 7 \}$$

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

$$[1 \ 4]: \{ 2 \ 3 \ 7 \ 8 \}$$

#Total subarrays:

$$ar[] = \{ \underset{4}{\downarrow} \underset{2}{\downarrow} \underset{3}{\downarrow} \underset{7}{\downarrow} \underset{8}{\downarrow} \}$$

} #Sub = 5

#Total subarrays len = 5

$$ar[] = \{ \underset{4}{\downarrow} \underset{2}{\downarrow} \underset{3}{\downarrow} \underset{7}{\downarrow} \underset{8}{\downarrow} \}$$

} #Sub = 4

#Total subarrays len = N

$$ar[] = \{ \underset{4}{\downarrow} \underset{2}{\downarrow} \underset{3}{\downarrow} \underset{7}{\downarrow} \underset{8}{\downarrow} \}$$

} #Sub = 3

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

$$ar[] = \{ \underset{4}{\downarrow} \underset{2}{\downarrow} \underset{3}{\downarrow} \underset{7}{\downarrow} \underset{8}{\downarrow} \}$$

} #Sub = 2

$$ar[] = \{ \underset{4}{\downarrow} \underset{2}{\downarrow} \underset{3}{\downarrow} \underset{7}{\downarrow} \underset{8}{\downarrow} \}$$

} #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] + " ");  
}
```

3

Q1: Print all Subarrays.

0 1 2 3

Eg: arr[4] = {3 7 2 9}

start ↑ end
 $\text{arr}[N] = \{0 1 2 \dots [i] i+1 i+2 \dots N-1\}$

#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}

```
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.

Ex: $\text{arr}[3] = \{3 \ 4 \ 2\}$

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> &arr) {

long tsum = 0;

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]

long sum = 0; //1 subarray sum

for (int k = i; k <= j; k++) {

sum = sum + arr[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: $O(N + N^2)$ = $O(N^2)$ SC: $O(N)$

int N = ar.size();

Generate pf

pf()

long pf[N], sum = 0;

generate all subarray sums

for (int i=0; i < N; i++) {

 sum = sum + ar[i];

 pf[i] = sum;

}

long tsum = 0;

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;

}

Idea: In Question, Calculate Sum of all we apply Contributin Technique
Contributin Technique = Add Contributin of each ele in final ans.

$$\text{Ex: arr[3]} = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 2 \end{bmatrix}$$

Subarrays:

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

Contributin:

$$\begin{array}{ll} \text{ele occurrence} & \text{contributin} \\ \hline 3 * 3 & = 9 \\ 4 * 4 & = 16 \\ 2 * 3 & = 6 \\ \hline \text{Total Sum} & = 31 \end{array}$$

$$\text{Ex: arr[4]} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 8 & -1 & 4 \end{bmatrix}$$

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}

Contributin:

$$\begin{array}{ll} \text{ele occurrence} & \text{contributin} \\ \hline 2 * 4 & = 8 \\ 8 * 6 & = 48 \\ -1 * 6 & = -6 \\ 4 * 4 & = 16 \\ \hline \text{Total Sum} & = 66 \end{array}$$

Catch:

To calculate contributin of element we need to get it's occurrence.

Occurrences: In this problem, it is no;f times arr[i] comes in subarray.

In how many subarrays a particular index i will be present

0 1 2 3 4 5

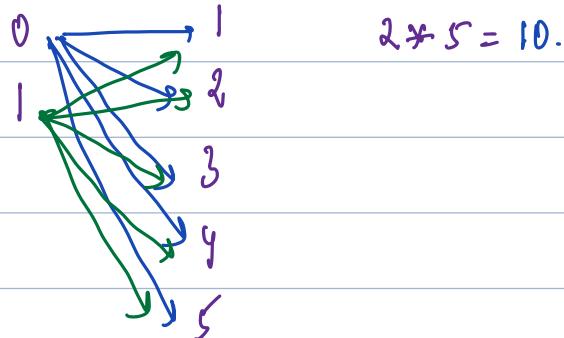
Ex: $ar[6] = \{3 -2 4 -1 2 6\}$ #Subarrays: {s..e}

#start = ✓ ✓ ✗ ✗ ✗ ✗

#end = ✗ ✓ ✓ ✓ ✓ ✓

In how many subarrays index i is present

startind endind Total Subarrays



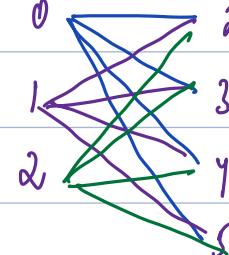
In how many subarrays index 2 is present

0 1 2 3 4 5

Ex: $ar[6] = \{3 -2 4 -1 2 6\}$ #startind endind Total Subarrays

#start = ✓ ✓ ✓ ✗ ✗ ✗

#end = ✗ ✗ ✓ ✓ ✓ ✓



In how many subarrays index i is present

$ar[N] = \{a_0 a_1 a_2 \dots a_{i-1} a_i a_{i+1} a_{i+2} \dots a_{N-2} a_{N-1}\}$ 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
$N=4$	$\text{arr}[y] = \{ 2 \}$	8	-1	4	
$\#(i+1) = 1$		2)	3)	4)	
$\#(N-i) = 4$		3	2	1	
#occurrence = 4		6	6	4	

$$\#\text{Contribut} = 8 + 48 - 6 + 16 = 66$$

long subSum(rectangle &ar) { TC: $O(N)$ SC: $O(1)$ }

```
int N = arr.size();
```

```
long sum = 0;
```

```
for (int i = 0; i < N; i++) {
    long occ = (i+1)[N-i];
    sum = sum + arr[i] * occ;
}
```

```
return sum;
```

3