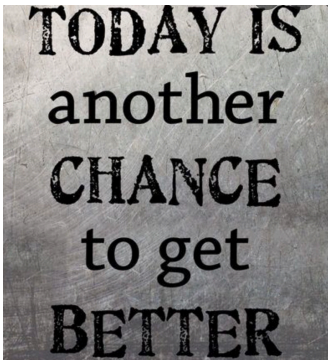
Today's Quote →



So, let's get better at sub-arrays.

Sunday (11/09/2022) → Problem Solving Session

- → problems given in assignment/homework which are least solved.
- → Idea & pseudo code discussed.
- → optional lecture, Attd. not counted
- → Recorded.

Recap.

Subarray → continuous part of an array.

- → single element / complete array → subarray
- → Empty array is not a subarray.
- → $i - j$: length $= j - i + 1$

Properties of Subarrays

arr[4] : { 2 6 3 9 }
0 1 2 3

count : 4 + 3 + 2 + 1 = 10.

subarrays :

[0, 0] : {2}
[0, 1] : {2, 6}
[0, 2] : {2, 6, 3}
[0, 3] : {2, 6, 3, 9}

[1, 1] : {6}
[1, 2] : {6, 3}
[1, 3] : {6, 3, 9}

[2, 2] : {3}
[2, 3] : {3, 9}

[3, 3] : {9}.

arr[N] : a_0 a_1 a_2 a_3 — — — — — — — — a_{N-2} a_{N-1}

Subarrays :

[0,0]	[1, 1]	[2,2]	[n-1, n-1]
[0,1]	[1,2]	[2,3]	
[0,2]	[1,3]	[2,4]	
[0,3]	⋮	[2,5]	
⋮		⋮	
[0, N-1]	[1, n-1]	[2, N-1]	

$\# \; N + (N-1) + (N-2) + \cdots + 1$

$$\boxed{\text{Total no. of sub-arrays} = \frac{N(N+1)}{2}.}$$

Problems :

① Given arr[N], [s, e] print subarray from si to ei.

start-index end-index

```
void printSubarray( arr, s, e){
    for( i -> si to ei){
        print( arr[i])
    }
}
```

$\boxed{T.C \rightarrow O(N)}$

arr[5] : { 1 2 13 9 14 20 6 }

1	2	13	9	14	20	6
0	1	2	3	4	5	6

$\boxed{\begin{matrix} si \rightarrow 2 \\ ei \rightarrow 5 \end{matrix}}$

1 subarray → N iterations

N^2 subarrays → $N^2 \times N$ iterations

→ N^3

Q) Given N array elements. print all the subarrays.

T.C → $O(N^3)$

arr[4] : { 6 8 -1 7 }
0 1 2 3

subarray

[0,0] : {6}
[0,1] : {6, 8}
[0,2] : {6, 8, -1}
[0,3] : {6, 8, -1, 7}
[1,1] : {8}
[1,2] : {8, -1}
[1,3] : {8, -1, 7}
[2,2] : {-1}
[2,3] : {-1, 7}
[3,3] : {7}

[0,0] [0,1] [0,2] [0,3]
[1,1] [1,2] [1,3]
[2,2] [2,3]
[3,3]

pseudo-code.

```
void printAllSubarrays ( arr, N) {
    for( i = 0 to n-1) {        → starting index of subarray
        for( j = i to n-1) {    → ending index of subarray.
            // print subarray from [s,e]
            for(k → i to j){print arr[k]}
            print (newLine)
        }
    }
```

T.C → $O(N^3)$
S.C → $O(1)$

Q) Given N array elements print each sub-array sum.

arr[4] : { 6 8 -1 7 }
0 1 2 3

Sub-array	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-1.

```
void printAllSums ( arr, N) {
    starting index of subarray
    for( i = 0 to n-1) {
        ending index of subarray.
        for( j = i to n-1) {
            // print sum of subarray [s,e]
            sum = 0
            for(k -> i to j) {sum += arr[k]}
            print(sum)
        }
    }
}
```

---

idea.2. : prefix. Sum.

```
void printSums ( arr, N) {
    // pSum [N]    // to do.
    starting index of subarray
    for( i = 0 to n-1) {
        ending index of subarray.
        for( j = i to n-1) {
            // sum of sub-array [i, j]
            if (i == 0)  print ( pSum[j] )
            else  print (pSum[j] - pSum[i-1])
        }
    }
}
```

T.C → $O(N^2)$
S.C → $O(N)$

You can't modify the array

Q) Given arr[N]. Print all sub-array sums starting at index 3.

arr[9] :	3	8	2	4	7	9	4	3	2
	0	1	2	3	4	5	6	7	8

subarray.	sum.
[3,3] :	4
[3,4] :	11
[3,5] :	20
[3,6] :	24
[3,7] :	27
[3,8] :	29

sum = 0+4

sum = 4+7

sum = 11+9

sum = 20+4

sum = 24+3

sum = 27+2

sum = 29

pseudo-code.

```
void printSums( arr, N) {
    sum = 0
    for ( j -> i to n-1) {
        sum = sum + arr[j]
        print(sum)
    }
}
```

T.C → O(N)
S.C → O(1)

final pseudo-code.

Printing all sub-array sums using carry forward.

```
void printSums( arr, N) {
        starting index of the sub-array
    for( i = 0 to n-1) {
        sum = 0
        for( j -> i to n-1) {
            sum = sum + arr[j]
            print(sum)
        }
    }
}
```

print all sub-array sums starting from ith index.

$\boxed{T.C \rightarrow O(N^2) \ , \ S.C \rightarrow O(1)}$

$i = 0$: print all sub-array sums starting from idx 0

$i = 1$: print all sub-array sums starting from idx 1

$i = 2$: print all sub-array sums starting from idx 2

⋮

$i = N-1$: print all sub-array sums starting from idx $N-1$

All sub-array sums are printed.

Q) Given N array elements, return sum of all subarray sums.

arr[4] : { 6 8 -1 7 }
0 1 2 3

subarray	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

Sum of all subarray sums = 94

arr[3] : { 4 3 7 }
0 1 2

subarray	sum
[0,0] :	4
[0,1] :	7
[0,2] :	14
[1,1] :	3
[1,2] :	10
[2,2] :	7

Sum of all subarray sums : 45

idea : For every subarray, get sum & add it total sum.

Approach 1.
→ 3 nested loops
→ $O(N^3)$, S.C: $O(1)$

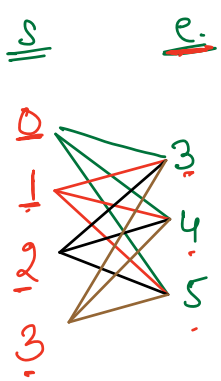
Approach 2.
→ prefix Sum
→ $O(N^2)$, SC: $O(N)$

Approach-3
→ Carry forward
→ $O(N^2)$, SC: $O(1)$

Final Solution

arr[6] : { 3 -2 4 -1 2 6 }
0 1 2 3 4 5

In how many sub-arrays index -3 is present?



$\boxed{4 * 3 = 12}$

arr[5] : { 3 -2 4 -1 2 }
0 1 2 3 4

In how many sub-arrays index-1 is present?

s	e
0	1
	2
1	3
	4

$\boxed{2 * 4 = 8}$

Generalize- Given n elements, # subarrays ith index is present.

arr[N] : $a_0 \quad a_1 \quad a_2 \quad --- \quad a_{i-1} \quad a_i \quad a_{i+1} \quad ---- \quad a_{N-2} \quad a_{N-1}$

s: [0 i] e: [i N-1]

$i+1$ X $n-i$

Eg: arr[4] : { 6 8 -1 7 } $\boxed{N=4}$

	6	8	-1	7
	0	1	2	3
i+1 :	1	2	3	4
N-i :	4	3	2	1
	4	6	6	4

individual contribution : 24 + 48 + (-6) + 28 = 94

Sum of all sub-array sums

Eg: arr[3] : { 4 3 7 } N=3.

	4	3	7
	0	1	2
i+1 :	1	2	3
N-i :	3	2	1
total :	3	4	3

Sum = (4*3) + (3*4) + (7x3)

= 45

Subarrays.

[0, 0] : { 4 }

[0, 1] : { 4, 3 }

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

[1, 1] : { 3 }

[1, 2] : { 3, 7 }

[2, 2] : { 7 }

(4x3) + (3x4) + (7x3)

pseudo-code.

```
int totalSum ( arr, n) {
    totalsum = 0
    for( i -> 0 to n-1) {
        //How many times idx i present in all subarrays
        contri = arr[i] * [(i+1) * (N-i)]
        totalsum += contri
    }
    return total sum
}
```

Contribution technique

$\boxed{T.C \rightarrow O(N) \quad , \quad S.C \rightarrow O(1)}$