Subarrays

Subarray → continous part of an array

→ sigle elements / complete array

→ empty [ ] is not subarray

→ $[i, j]$: length $= j - i + 1$

What are total possible subarrays ?

$a[4] = \underset{0}{2} \quad \underset{1}{6} \quad \underset{2}{3} \quad \underset{3}{9}$

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

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

[2,2]
[2,3]
2

[3,3]
1

$= 4+3+2+1 = 10$

What are total subarrays for array of size N ?

$a[n] = \underset{0}{a_0} \quad \underset{1}{a_1} \quad \underset{2}{a_2} \quad \ldots\ldots \quad \underset{n-2}{a_{n-2}} \quad \underset{n-1}{a_{n-1}}$

[0,0]
[0,1]
[0,2]
⋮
[0, n-1]

count = n-1-0+1 = n

[1,1]
[1,2]
⋮
[1, n-1]

count = n-1

....

[n-2, n-2]
[n-2, n-1]

2

[n-1, n-1]

1

total subarrays = $n + n-1 + n-2 + \dots + 2 + 1$

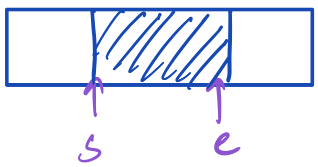
$$= \boxed{\frac{n(n+1)}{2}}$$

Question 1

Given a[N], s & e integers. $0 <= s, e < n$

Print subarray from [s, e] $s <= e$

a[n] =



eg a[5] =

$$\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ 2 & 4 & 5 & 9 & 8 \\ & \uparrow & & \uparrow & \\ & s=1 & & e=3 & \end{array}$$

ans = 4 5 9

Code

```
for (i=s; i<=e; ++i)
    print(a[i])
```

Question 2

Given N array elements, print each & every subarray.

Note: Do it without extra space.

eg a[4] =

$$\begin{array}{cccc} 6 & 8 & -1 & 7 \\ 0 & 1 & 2 & 3 \end{array}$$

[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}

total subarrays $= \frac{N(N+1)}{2}$

$\approx O(N^2)$

I have to print $O(N^2)$ lines.

worst case to print subarray $= O(N)$

Time to take to print all subarrays

$\Rightarrow O(N^2) * O(N)$

$\Rightarrow O(N^3)$

Code

```
def printAll( a[]) {
    n = a.length
    for (i=0; i<n; ++i) {
        for (j=i; j<n; ++j) {
            // [i,j] subarray
            for (K=i ; K<=j ; ++K )
                print (a[K])

            print(newline)
        }
    }
}
```

TC: $O(N^3)$

SC : $O(1)$

Question 3

Given N array elements, print each subarray sum.

eg a[4] = 6 8 -1 7

0 1 2 3

[0,0] → 6

[0,1] → 6+8 = 14

[0,2] → 6+8+-1 = 13

[0,3] → 20

[1,1] → 8

[1,2] → 7

[1,3] → 14

[2,2] → -1

[2,3] → 6

[3,3] → 7

print $O(N^2)$ values

```
def printsubarraysum(a[]) {
    n = a.length
    for (i=0; i<n; ++i) {
        for (j=i; j<n; ++j) {
            // [i,j] subarray
            sum=0
            for (k=i; k<=j; ++k)
                sum = sum + a[k]
```

TC: $O(N^3)$

SC: $O(1)$

```
            print( sum )
        }
      }
  }
```

⇓ OPTIMIZE
using Prefix Sum

code

```
void printSubarraySum( a[] ) {
        n = a.length                          TC        SC
        pf[n] ⟶ TODO          →    O(N) ,    O(N)
        for (i=0; i<n; ++i ) {            ]
            for (j=i; j<n; ++j ) {        ]→ O(N^2), O(1)
                // [i,j] subarray
                if(i==0)      print (pf[j])
                else          print( pf[j] - pf[i-1])
            }
        }
}
```

TC : $O(N^2)$

SC : $O(N)$

Question 4

Given a[N], print all subarray sums starting at index 3.

eg a[8] =

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

Sum
7
16
20
23
25

Carry forward

sum = 0+7
sum = 7 + 9
sum = 16 + 4
sum= 20 + 3
sum = 23 + 2

```
sum = 0
for (j = 3; j<n; ++j) {
    sum = sum + a[j]
    print (sum)
}
```

Question 4 answer

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

TC : $O(N^2)$
SC : $O(1)$

Optimized answer of Question 3

Dry run   a[4] = 6 8 -1 7
                   0 1   2 3

i=0   sum=0   j= [0, 3]

sum = 0 + a[0] = 6   print(6)

Sum = 6 + a[1] = 14   14

sum = 14 + a[2] = 13   13

sum = 13 + a[3] = 20   20

i=1   sum=0   j = [1, 3]

sum = 0 + a[1] = 8   8

⋮   ⋮

i=2   ⋮

i=3

Question 5

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

A[4] = 6 8 -1 7
        0 1   2 3

[0,0] → 6

[0,1] → 6+8 = 14

[0,2] → 6+8+-1 = 13

[0,3] → 20

[1,1] → 8

[1,2] → 7

[1,3] → 14

[2,2] → -1

[2,3] → 6

[3,3] → 7

ans = 6 + 14 + 13 + 20
+ 8 + 7 + 14
+ -1 + 6
+ 7

⇒ 94

Idea: for every subarray get sum & add it to total sum.

Approach 1
→ 3 nested loops
$O(N^3)$ $O(1)$

Approach 2
→ prefix sum
$O(N^2)$ $O(N)$

Approach 3
→ carry forward
$O(N^2)$ $O(1)$

Code

```
ans=0
for(i=0; i<n; ++i) {
    sum=0
    for(j=i; j<n; ++j) {
        sum=sum+a[j]
        ans = ans +sum
    }
}
return ans
```

TC: $O(N^2)$

SC: $O(1)$

eg A[6] = 3 -2 4 -1 2 6

0 1 2 3 4 5

In how many subarrays, index 3 is present?

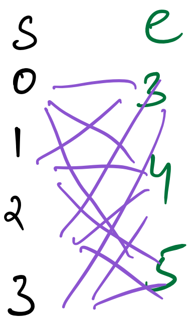
[0,3] [0,4] [0,5]

[1,3] [1,4] [1,5]

[2,3] [2,4] [2,5]

[3,3] [3,4] [3,5]

total=12

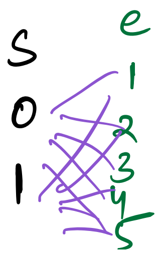


⇒ 4×3 =12

In how many subarrays, index 1 is present?

[0,1] [0,2] [0,3] [0,4] [0,5]

[1,1] [1,2] [1,3] [1,4] [1,5]



⇒ 2×5=10

Generalize: Given N elements, find number of subarrays where i^{th} index is present.

$$A[N] = a_0 \; a_1 \; a_2 \; \cdots \; a_{i-1} \; a_i \; a_{i+1} \; \cdots \; a_{n-2} \; a_{n-1}$$

$s : [0, i]$

$e : [i, n-1]$

$\# \; i-0+1 = i+1$

$\# \; n-1-i+1$
$= n-i$

$\Rightarrow (i+1) \times (n-i)$

$A[4] = 6 \quad 8 \quad -1 \quad 7$ $\qquad n=4$

		6	8	-1	7
		0	1	2	3
$i+1$	=	1 ×	2	3	4
$n-i$	=	4	3	2	1
$(i+1)\times(n-i)$	=	4	6	6	4

individual contribution = 24 + 48 + -6 + 28 = 94

N=3

a[3]	=	4	3	7
		0	1	2
i+1	=	1	2	3
n-i	=	3	2	1
(i+1)*(n-i)	=	3	4	3

sum = 12 + 12 + 21

= 45

subarrays

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

Code

```
ans=0
for(i=0; i<n; ++i) {
    total = (i+1)x(n-i)
    contribution = total x a[i];
    ans = ans + contribution
}
return ans
```

TC: O(N)

SC: O(1)