Subarrays

Subarray → continuous part of an array

→ single elements / complete array

→ empty {} is not subarray

→ [i, j] : length = j - i + 1

↳ {i, i+1, i+2, ... j}

What are total possible subarrays ?

a[4] = 2 6 3 9

0 1 2 3

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

= 4+3+2+1 = 10

What are total subarrays for array of size N?

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

$0 \quad 1 \quad 2 \quad n-2 \quad n-1$

[0,0]
[0,1]
[0,2]
.
.
.
:
[0, n-1]

count = n-1 -0 +1 = n

[a,b] = b-a+1

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

count = n-1 - 1+1 = n-1

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

2

[n-1, n-1]

1

total subarrays = n + n-1 + . - - . + 2 + 1

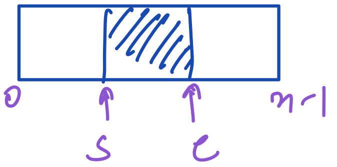
$$= \frac{n(n+1)}{2}$$

Question 1

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

Print subarray from [s, e]

a[n] =



eg a[5] =

0	1	2	3	4
2	4	5	9	8
	s=1		e=3	

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] =

0	1	2	3
6	8	-1	7

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

$\approx O(N^2)$

To print all subarrays, I have to print $O(N^2)$ lines

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

Time taken to print all subarrays

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

$= O(N^3)$

Code

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

```
for(k=i; k<=j; ++k)
    print(a[k])
```

Question 3

Given N array elements, print each subarray sum.

eg a[4] = 6 8 -1 7

6	8	-1	7
0	1	2	3

[0,0] → 6

[0,1] → 6+8 = 14

[0,2] → 6+8+-1 = 13

[0,3] → 6+8+-1+7 = 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 printSum( 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 += a[K]
            print(sum)
        }
    }
```

TC: $O(N^3)$

SC: $O(1)$

⇓ OPTIMISE using Prefix Sum

Code

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

→ TC: $O(N^2)$

SC: $O(1)$

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
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

25

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

Question 4 code

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

TC : $O(N^2)$

SC : $O(1)$

Optimized code 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

sum = 8 + a[2] = 7 7

⋮

i = 2

i = 3

BREAK : 8:05 - 8:15

Question 5

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

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] → 6+8+-1+7 = 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

print O(1) values.

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 += a[j]
        ans += sum
    }
}
print(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?

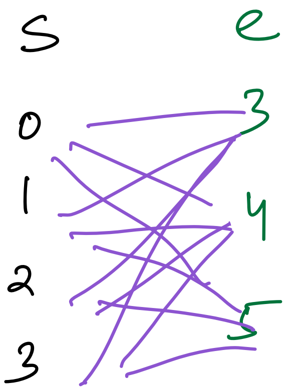
[0,3] [0,4] [0,5]

[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?

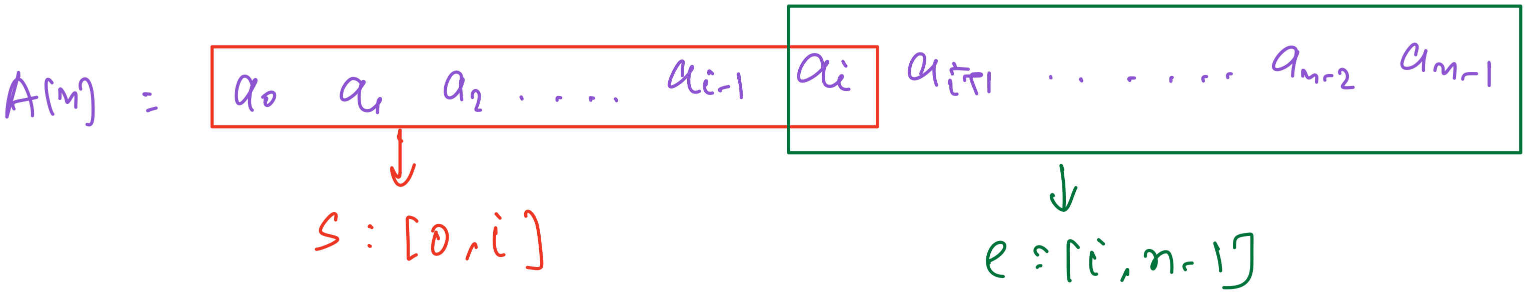
[0,1] [0,2] [0,3] [0,4] [0,5]

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

s	e
0	1
1	2
	3
	4
	5

=> 2×5 = 10

Generalize : Given N elements, find number of subarrays where ith index is present?


A[N] =
a_0 a_1 a_2 . . . . a_{i-1} a_i a_{i+1} . . . . a_{n-2} a_{n-1}
s : [0, i]
e : [i, n-1]

count = $i - 0 + 1$

$= i + 1$

count = $n - 1 - i + 1$

$= n - i$

total = $(i+1) * (n-i)$

		0	1	2	3
a[4]	=	6	8	-1	7
i+1	:	1	2	3	4
n-i	:	4	3	2	1
(i+1)*(n-i)	:	4	6	6	4

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

Dry run

N=3

a[3]	=	4	3	7
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)*(n-i)
    ans += (total * a[i])
}
print(ans)
```

TC: O(N)

SC: O(1)