Arrays - Subarrays

1. Continuous part of an array is called Subarray.
2. A single element is a Subarray.
3. Entire array is a Subarray.
4. Empty cannot be a Subarray.

	0	1	2	3	4	5	6	7	8
ar[9] =	-3	4	6	2	8	7	14	9	21

Ex 1: indices [2, 3, 4] ✓

Ex 2: indices [0, 1, 3, 4] ✗

Ex 3: indices [6] ✓

Ex 4: indices [0, 5] ✗

Ex 5: subarray [0 4]
start ↑ ↑ end

0	1	2	3	4
-3	4	6	2	8

$0 \leq \text{start} \leq \text{end} \leq n-1$

Count of subarrays

0	1	2	3	4	5	6
4	2	10	3	12	-2	15

Subarray starting from 0th index :

[0 0], [0 1], [0,2], [0,3], [0 4],
[0 5], [0 6] ⇒ 7

Subarray starting from 1st index : Quiz 1

[1 1], [1 2], [1 3], [1 4],
[1 5], [1 6] ⇒ 6

Given N array elements, how many subarrays can be generated ?

ar[N] = [0 1 2 3 ..... i ith ..... N-2 N-1]

Start 0	Start 1	Start 2	... Start N-2	Start N-1
[0 0]	[1 1]	[2 2]	[N-2 N-2]	
[0 1]	[1 2]	[2 3]	[N-2 N-1]	
[0 2]	[1 3]	[2 4]		
⋮	⋮	⋮		
[0 N-1]	[1 N-1]	[2 N-1]	[N-1 N-1]	
N	N-1	N-2	2	1

Quiz 2

Total = N + (N-1) + (N-2) + ... + 2 + 1

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

Q1. Print all values of a subarray

array, start, end

```
printSubarray(A[ ], s, e) {

    for( i=s; i<=e ; i++ ) {
        print (A[i])
    }

}
```

Time - O(N)
Space - O(1)

Q2. Find the sum of all elements in a given subarray.

```
addSubarray(A[ ], s, e) {

    sum = 0
    for( i=s; i<=e; i++) {
        sum += A[i]
    }

    return sum

}
```

Time - O(N)
Space - O(1)

Q3. Print all subarrays of a given array.

A : 2 8 9 (indices 0 1 2)

[0 0] : 2

[0 1] : 2 8

[0 2] : 2 8 9

[1 1] : 8

[1 2] : 8 9

[2 2] : 9

N=4 [0 1 2 3]

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

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

[2 2] [2 3]

[3 3]

i → [0, n-1]

j → [i, n-1]

Quiz 3

```
for ( i=0; i<N; i++) {          <- N
    for( j=i; j<N; j++) {       <- N
        // subarray [i j]
        print subarray (A, i, j)   <- N
    }
}
```

TC : O(N^3)

SC : O(1)

No of subarrays $= \frac{N(N+1)}{2} \approx N^2$

Java

```java
void printSubarray(int []A, int start, int end) {
    for (int i = start; i <= end; i++) {
        System.out.print(A[i] + " ");
    }
    System.out.println();
}

void printAllSubarrays(int []A) {
    int n = A.length;
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            printSubarray(A, i, j);
        }
    }
}
```

Python

```python
def printSubarray(A, start, end):
    for i in range(start, end + 1):
        print(A[i], end=" ")
    print()


def printAllSubarrays(A):
    n = len(A)

    for i in range(n):
        for j in range(i, n):
            printSubarray(A, i, j)
```

Q4. Print sum of every single subarray.

	0	1	2
A =	3	-2	4

[0 0] : 3

[0 1] : 1

[0 2] : 5

[1 1] : -2

[1 2] : 2

[2 2] : 4

```
SumOfSubarrays(int []A, int s, int e) {
    for ( i=0;  i<N;  i++ ) {          <- N
        for( j=i; j<N; j++ ) {         <- N
            // subarray [i  j]
            sum = addSubarray ( A, i, j )
            print(sum)
        }
    }
}
```

TC: $O(N^3)$

Java

```java
int addSubarray(int[] A, int start, int end) {
    int s = 0;
    for (int i = start; i <= end; i++) {
        s += A[i];
    }
    return s;
}

void subarraySumBruteForce(int[] A) {
    int n = A.length;
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            int s = addSubarray(A, i, j);
            System.out.println(s);
        }
    }
}
```

Python

```python
def addSubarray(A, start, end):
    s = 0
    for i in range(start, end + 1):
        s += A[i]
    return s


def subarraySumBruteForce(A):
    n = len(A)
    for i in range(n):
        for j in range(i, n):
            s = addSubarray(A, i, j)
            print(s)
```

Time - O(N^3)

Space - O(1)

Optimisation

addSubarray (A, i, j) → sum [i j] O(N) time

$$sum[L \quad R] = \begin{cases} \text{if } L == 0: & pf[R] \\ \text{else}: & pf[R] - pf[L-1] \end{cases}$$

```
SumOfSubarrays(int []A, int s, int e) {
    // Create prefix sum array  <- TODO <- O(N) time
    for(i=0; i<N; i++) {
        for(j=i; j<N; j++) {
            // sum[i j]
            if (i==0)
                sum = pf[j]
            else
                sum = pf[j]-pf[i-1]
            print(sum)
        }
    }
}
```

Extra space for pf [N]

Time - O(N^2)
Space - O(N)

Q5. Print sum of all the subarrays starting from index 2.
Note: The given array must not be modified.

Expected SC: O(1)

	0	1	2	3	4	5	6	7
arr[7] = [	7	3	2	-1	6	8	2	5]

[2 2]: 2

[2 3]: 1

[2 4]: 7

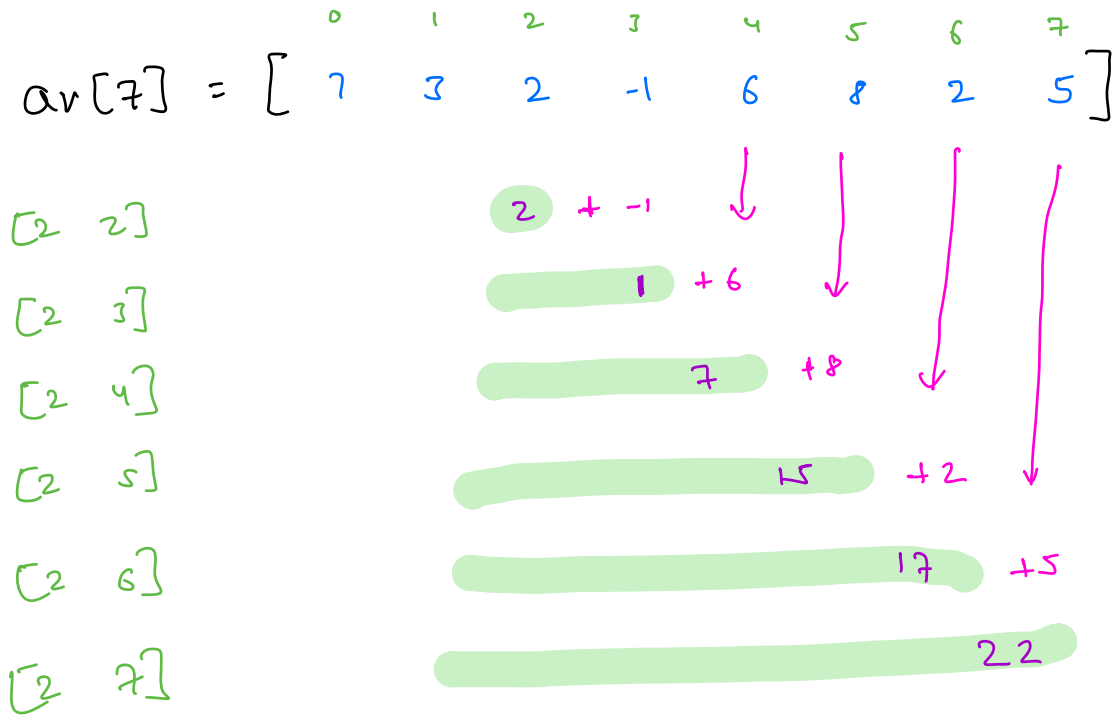
[2 5]: 15

[2 6]: 17

[2 7]: 22

```
for(j=2; j<N; j++) {
    s = addSubarray(A, 2, j)
    print(s)
}
```

Time = $O(N^2)$



Carry forward → Sum of previous subarray

From L to R

```
Sum = 0
for (j = 2; j < n; j++) {
    sum = sum + A[j]
    print (sum)
}
```

TC: O(N)

SC: O(1)

Q6 Print all subarray sum starting at index = 3

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

Q7 Print all subarray sum starting at index = i

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

Q8 What will this code do ?

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

Print all subarray sums using carry forward

TC: $O(N^2)$

SC: $O(1)$

Java

```java
void sumOfAllSubarray(int []A) {
    int n = A.length;
    for (int i = 0; i < n; i++) {
        int sum = 0;
        for (int j = i; j < n; j++) {
            sum += A[j];
            System.out.println(sum);
        }
    }
}
```

Python

```python
def sumOfAllSubarray(A):
    n = len(A)
    for i in range(n):
        s = 0
        for j in range(i, n):
            s += A[j]
            print(s)
```

Time - $O(N^2)$

Space - $O(1)$

Break till 10 : 16 PM

Q9 Given an array, find sum of all subarray sums.

Expected SC: O(1)

	0	1	2
A :	3	-1	4

s	e	sum
[0	0]	3
[0	1]	2
[0	2]	6
[1	1]	-1
[1	2]	3
[2	2]	4
		17

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

Time - $O(N^2)$

Space - O(1)

	0	1	2
A :	3	-1	4

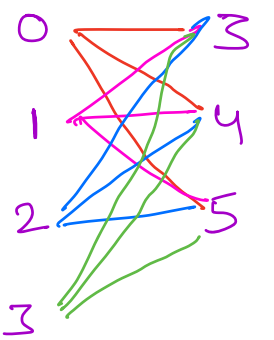
s	e	sum	
[0	0]	3	3
[0	1]	2	3 + -1
[0	2]	6	3 + -1 + 4
[1	1]	-1	-1
[1	2]	3	-1 + 4
[2	2]	4	4
		17	3 x 3 + -1 x 4 + 4 x 3

If we know in how many subarrays, each element is coming, can we solve this faster?

Q10. In how many subarrays index 3 is present ?

	0	1	2	3	4	5
A =	3	-2	4	-1	2	6

start end



4 x 3 ⟶ Count = 12

Q11. In how many subarrays index 1 is present ? Quiz 4

	0	1	2	3	4	5
A =	3	-2	4	-1	2	6

start	end
0	1
1	2
	3
	4
	5

$2 \times 5 = 10$ pairs

$= 10$ subarrays

Q11. Given arr[N], in how many subarrays index i is present ?

Quiz 5

$A = a_0 \quad a_1 \quad a_2 \quad a_3 \quad \ldots \quad a_{i-1} \quad a_i \quad a_{i+1} \quad \ldots \quad a_{n-1}$

start : $[0 \quad i] \Rightarrow (i+1)$ elements

end : $[i \quad N-1] \Rightarrow (N-i)$ elements

In how many subarrays a_i is coming $\Rightarrow (i+1) \times (N-i)$

Back to Q9

	0	1	2	3	4	5
A =	3	-2	4	-1	2	6
start : i+1	1	2	3	4	5	6
end : N-i	6	5	4	3	2	1
Count (i+1)*(N-i)	6	10	12	12	10	6

Total Sum = $3 \times 6 + -2 \times 10 + 4 \times 12 + -1 \times 12 + 2 \times 10 + 6 \times 6$

$= 18 + -20 + 48 + -12 + 20 + 36$

$= 90$

Idea:

Adding contribution of each A[i] element in the total sum

Contribution Technique

TC: O(N)

SC: O(1)

```
ans = 0
for (i=0; i<n; i++) {
    count = (i+1) * (N-i)
    ans += (count * A[i])
}
```

Java

```java
int contributionSum(int[] A) {
    int n = A.length;
    int s = 0;

    for (int i = 0; i < n; i++) {
        int c = (i + 1) * (n - i);
        s = s + (A[i] * c);
    }

    return s;
}
```

Python

```python
def contributionSum(A):
    n = len(A)
    s = 0
    for i in range(n):
        c = (i + 1) * (n - i)
        s = s + (A[i] * c)
    return s
```

Time - O(N)

Space - O(1)

Doubts

Thank You

No of subarrays = $\frac{N(N+1)}{2}$

Print each subarray = $O(N)$

Total = $\frac{N(N+1)}{2} \times N = \frac{N^2(N+1)}{2}$

$= O(N^3)$

2^N

0	1	2	3
1	9	2	8

pfodd = 0 9 → 9 17

pfodd[i] = sumodd[0 i]

Good Night

Thank You

Monday