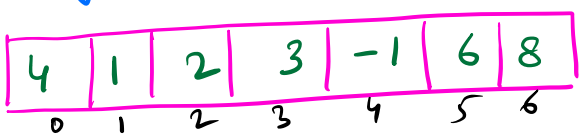
Act as if what you do makes a difference. It does!

— William James

Subarrays

A subarray is a contiguous part of an array.



1 2 3 → ✓

1 3 2 → ✗

3 -1 8 → ✗

4 1 2 3 -1 6 8 → ✓

6 → ✓

→ single element is a subarray.

→ whole array is also a subarray.

(start, end) → indices that uniquely represent the subarray.

How many subarrays (non-empty) do we have in an array of size n?

$$\frac{n(n+1)}{2} \equiv O(n^2)$$

```
void printSubarray(int arr[], int start, int end){
    for(int i = start; i<=end; i++){
        print(arr[i]);
    }
}

int sumSubarray(int arr[], int start, int end){
        int sum = 0;
        for(int i = start; i<=end; i++){
            sum+=arr[i];
        }
        return sum;
}
```

Q1) Print all possible subarrays

A → {2 8 9}

s	e	
0	0	→ 2
0	1	→ 2 8
0	2	→ 2 8 9
1	1	→ 8
1	2	→ 8 9
2	2	→ 9

```
// i denotes the start index
for(int i=0; i<n; i++){
    // j denotes the end index
    for(int j=i; j<n; j++){
        (i,j) ≡ (start, end) → 1 subarray
        for(int k=i; k<=j; k++){
            print(arr[k]);
        }
        print("\n");
    }
}
```

$O(n^3)$

T.C.

$O(1)$

S.C.

We cannot do better!

```
// i denotes the start index
for(int i=0; i<n; i++){
    // j denotes the end index
    for(int j=i; j<n; j++){
        (i,j) ≡ (start, end)
        for(int k=i; k<=j; k++){
            print(arr[k]);
        }
        print("\n");
    }
}
```

n=3
{2, 8, 9}

i	j	k	
0	0	0	[0,0]
0	1	0 1	[0,1]
0	2	0 1 2	[0,2]
1	1	1	[1,1]
1	2	1 2	[1,2]
2	2	2	[2,2]

Output

```
2
2 8
2 8 9
8
8 9
9
```

Q2) Find the sum of each possible subarray

A → {2 8 9}

s	e	
0	0	→ 2
0	1	→ 10
0	2	→ 19
1	1	→ 8
1	2	→ 17
2	2	→ 9

```
int ans[] = new int[n*(n+1)/2];
int c=0;
// i denotes the start index
for(int i=0; i<n; i++){
    // j denotes the end index
    for(int j=i; j<n; j++){
        (i,j) ≡ (start, end) → 1 subarray
        int sum=0;
        for(int k=i; k<=j; k++){
            sum+=arr[k];
        }
        ans[c] = sum;
        c++;
    }
}
return ans;
```

$O(n^3)$ T.C

$O(n^2)$ S.C.

$O(1)$ if we print.

```
// i denotes the start index
for(int i=0; i<n; i++){
    // j denotes the end index
    for(int j=i; j<n; j++){
        (i,j) ≡ (start, end)
        int sum=0;
        for(int k=i; k<=j; k++){
            sum+=arr[k];
        }
        ans[c] = sum;
        c++;
    }
}
```

{2,8,9}

i	j	k		Output
0	0	[0,0]	0+2	= 2
0	1	[0,1]	0+2+8	= 10
0	2	[0,2]	0+2+8+9	= 19
1	1	[1,1]	0+8	= 8
1	2	[1,2]	0+8+9	= 17
2	2	[2,2]	0+9	= 9

[Break till 10:43 PM]

$sum[i,j] \longrightarrow sum[i,j-1] + ar[j]$

```
// i denotes the start index
for( int i=0; i<n; i++){
    // j denotes the end index
    int sum = 0;
    for(int j = i; j<n; j++){
        (i,j) = (start, end)
        sum += arr[j];
        print(sum);
    }
}
```

$O(n^2)$ T.C.,

$O(1)$ S.C.

$\{2, 8, 9\}$

$i=0$

sum = 0

$j=0$ ⇒ sum = 0 + arr[j]
= 0 + arr[0]
= 2

$j=1$ ⇒ sum = 2 + arr[1]
= 10

$j=2$ ⇒ sum = 10 + 9
= 19

$i=1$

sum = 0

$j=1$ ⇒ sum = 0 + 8
= 8

$j=2$ ⇒ sum = 8 + 9 = 17

$i=2$

sum = 0

$j=2$ ⇒ sum = 0 + 9
= 9

Q 3) Find the sum of all subarray sums.

Google
Meta

A → {2 8 9}

s	e	
0	0	→ 2
0	1	→ 10
0	2	→ 19
1	1	→ 8
1	2	→ 17
2	2	→ 9

65.

```
int total = 0;
// i denotes the start index
for( int i=0; i<n; i++){
    //j denotes the end index
    int sum = 0;
    for(int j=i; j<n; j++){
        (i,j) = (start, end)
        sum += arr[j];
        total += sum;
    }
}
return total;
```

$O(n^2)$ T.C.
$O(1)$ S.C.

A → {-1 5 8} (x y z)

[0,0] →	-1	= -1
[0,1] →	-1 + 5	= 4
[0,2] →	-1 + 5 + 8	= 12
[1,1] →	5	= 5
[1,2] →	5 + 8	= 13
[2,2] →	8	= 8
	3*(-1) + 4*5 + 3*8	41
	= 41	

Ans → x*(-1) + y*5 + z*8

$3*(-1) + 4*5 + 3*8$

contribution of (-1) | contribution of 5 | contribution of 8

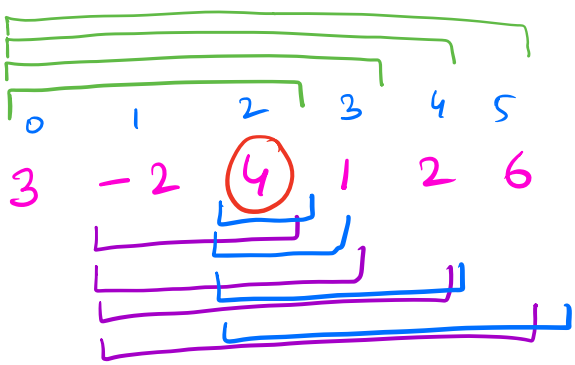
A → {3 -1 4 2}

no. of subarrays → 4 6 6 4 → How to find this?

$(3*4)+(-1*6)+(4*6)+(2*4)$

$= 38$

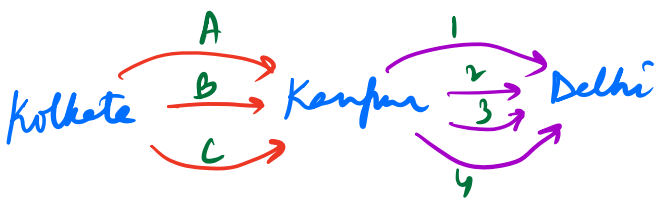
⇒ We can Hope that O(n) can be aimed!

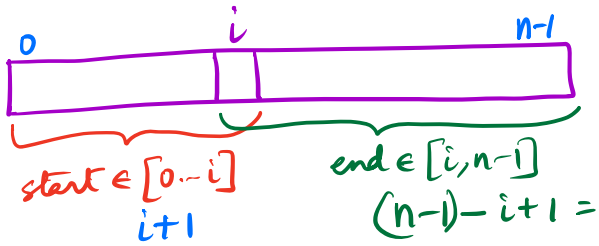


[Contribution Technique]

start → [0,1,2]
end → [2,3,4,5]

$3 * 4 = 12$





[a, b]
↓
b - a + 1

no. of subarrays (i^{th}) = $(i+1) * (n-i)$

```
int sum = 0;
for(int i = 0; i < n; i++){
    sum += a[i] * (i+1) * (n-i);
}
return sum;
```