1. Prefix Sum Formula –

🡪 sum(s, e) = pf[e] – pf[s-1];

2. Time Complexity of Prefix Sum –

🡪 to create a prefix sum array we iterate the array once so the time complexity to create a prefix sum array is O(N).

3. Range Sum Query –

**Problem Description**

You are given an integer array **A** of length **N**.  
You are also given a 2D integer array **B** with dimensions **M x 2**, where each row denotes a [L, R] query.  
For each query, you have to find the sum of all elements from L to R indices in A (0 - indexed).  
More formally, find A[L] + A[L + 1] + A[L + 2] +... + A[R - 1] + A[R] for each query.

**Problem Constraints**

1 <= N, M <= 105  
1 <= A[i] <= 109  
0 <= L <= R < N

**Input Format**

The first argument is the integer array A.  
The second argument is the 2D integer array B.

**Output Format**

Return an integer array of length M where ith element is the answer for ith query in B.

**Example Input**

Input 1:

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

B = [[0, 3], [1, 2]]

Input 2:

A = [2, 2, 2]

B = [[0, 0], [1, 2]]

**Example Output**

Output 1:

[10, 5]

Output 2:

[2, 4]

**Example Explanation**

Explanation 1:

The sum of all elements of A[0 ... 3] = 1 + 2 + 3 + 4 = 10.

The sum of all elements of A[1 ... 2] = 2 + 3 = 5.

Explanation 2:

The sum of all elements of A[0 ... 0] = 2 = 2.

The sum of all elements of A[1 ... 2] = 2 + 2 = 4.

Solution –

[[0, 3], [1, 2]] 🡪 representing this as matrix



|  |  |
| --- | --- |
| 0 | 3 |
| 1 | 2 |



Case 1 -

Idx 0,0 is the start and idx 0,1 is the end. So s=0 and e=3

Case 2 –

Idx 1,0 is the start and idx 1,1 is the end. So s=1 and j=2

Class Test{

Main(String[] args){

// calculate prefix sum

Int[] prefix = new int[arr.length];

Prefix[0] = arr[0];

For(int i=1;i<n; i++){

Prefix[i] = prefix[i-1] + arr[i];

}

For(int i=0; i<ar.length; i++){

For(int j=0; j<ar[0].length; j++){

Int start = ar[i][j];

Int end = ar[i][j];

}

}

}

}