Count 1s in a Sorted Binary Array

Given a binary array **arr[]** of size **N,** which is sorted in **non-decreasing order**, count number of **1's** in it.

Examples:

> ***Input:*** *arr[] = {0,0, 0, 0, 0, 1, 1}*
> ***Output:*** *2*
>
> ***Input:*** *arr[] = {1, 1, 1, 1, 1, 1, 1}*
> ***Output:*** *7*
>
> ***Input:*** *arr[] = {0, 0, 0, 0, 0, 0, 0}*
> ***Output:*** *0*

Method: Binary Search

C++ | Java

```
#include <iostream>
using namespace std;

int countOnes(int arr[], int n)
{
    int low = 0, high = n - 1;

    while(low <= high)
    {
        int mid = (low + high) / 2;

        if(arr[mid] == 0)
            low = mid + 1;
        else
        {
            if(mid == 0 || arr[mid - 1] == 0)
```

```
                high = mid -1;
    return 0;
}

int main() {

   int arr[] = {0, 0, 1, 1, 1, 1}, n = 6;

   cout << countOnes(arr, n);

    return 0;
}
```


Output

```
4
```

Time complexity: O(Log(N))
Auxiliary Space: O(1)