Count Inversions

Given an array of integers. Find the Inversion Count in the array.

Inversion Count*:* For an array, inversion count indicates how far (or close) the array is from being sorted. If array is already sorted then the inversion count is 0. If an array is sorted in the reverse order then the inversion count is the maximum.
Formally, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j.

Example 1:

Input: N = 5, arr[] = {2, 4, 1, 3, 5}
Output: 3
Explanation: The sequence 2, 4, 1, 3, 5
has three inversions (2, 1), (4, 1), (4, 3).

Example 2:

Input: N = 5
arr[] = {2, 3, 4, 5, 6}
Output: 0
Explanation: As the sequence is already
sorted so there is no inversion count.

Example 3:

Input: N = 3, arr[] = {10, 10, 10}
Output: 0
Explanation: As all the elements of array
are same, so there is no inversion count.

Your Task:
You don't need to read input or print anything. Your task is to complete the function **inversionCount()** which takes the array arr[] and the size of the array as inputs and returns the inversion count of the given array.
**Expected Time Complexity: **O(NLogN).
**Expected Auxiliary Space: **O(N).

```
class Solution:
    #User function Template for python3

    # arr[]: Input Array
    # N : Size of the Array arr[]
    #Function to count inversions in the array.
    def inversionCount(self, arr, n):
        # Your Code Here
        return self.mergeSort(arr,0,n-1)[1]

    def mergeSort(self,arr,lo,hi):
        if lo==hi:
            return [arr[lo]],0
        inv = 0
        mid = (lo+hi)//2
        left,leftInv = self.mergeSort(arr,lo,mid)
        right,rightInv = self.mergeSort(arr,mid+1,hi)
        final,finalInv = self.merge(left,right)
        return final,leftInv+rightInv+finalInv
    def merge(self,leftArr,rightArr):
        n = len(leftArr)
        m = len(rightArr)
        res = [0]*(n+m)
        inv = 0
        i = 0
        j = 0
        k = 0
        while i<n and j<m:
            if leftArr[i]<=rightArr[j]:
                res[k] = leftArr[i]
                k = k+1
                i = i+1
            else:
                res[k] = rightArr[j]
                inv = inv+(n-i)
                k = k+1
                j = j+1
        while i<n:
            res[k] = leftArr[i]
            i = i+1
            k = k+1
        while j<m:
            res[k] = rightArr[j]
            j = j+1
            k = k+1

        return res,inv
```