```
## Function to Use Inversion Count

def mergeSort(arr, n):

    # A temp_arr is created to store

    # sorted array in merge function

    temp_arr = [0]*n

    return _mergeSort(arr, temp_arr, 0, n-1)


## This Function will use MergeSort to count inversions



def _mergeSort(arr, temp_arr, left, right):



    # A variable inv_count is used to store

    # inversion counts in each recursive call



    inv_count = 0



    # We will make a recursive call if and only if

    # we have more than one elements



    if left < right:



        # mid is calculated to divide the array into two subarrays

        # Floor division is must in case of python



        mid = (left + right)//2



        # It will calculate inversion

        # counts in the left subarray



        inv_count += _mergeSort(arr, temp_arr,

                                    left, mid)



        # It will calculate inversion

        # counts in right subarray



        inv_count += _mergeSort(arr, temp_arr,

                                  mid + 1, right)



        # It will merge two subarrays in

        # a sorted subarray



        inv_count += merge(arr, temp_arr, left, mid, right)

    return inv_count


## This function will merge two subarrays
# in a single sorted subarray

def merge(arr, temp_arr, left, mid, right):

    i = left     # Starting index of left subarray

    j = mid + 1 # Starting index of right subarray

    k = left     # Starting index of to be sorted subarray

    inv_count = 0



    # Conditions are checked to make sure that

    # i and j don't exceed their

    # subarray limits.



    while i <= mid and j <= right:



        # There will be no inversion if arr[i] <= arr[j]



        if arr[i] <= arr[j]:

            temp_arr[k] = arr[i]

            k += 1

            i += 1

        else:

            # Inversion will occur.

            temp_arr[k] = arr[j]

            inv_count += (mid-i + 1)

            k += 1

            j += 1



    # Copy the remaining elements of left

    # subarray into temporary array

    while i <= mid:

        temp_arr[k] = arr[i]

        k += 1

        i += 1



    # Copy the remaining elements of right

    # subarray into temporary array

    while j <= right:
```

```
    while j <= right:

        temp_arr[k] = arr[j]

        k += 1

        j += 1



    # Copy the sorted subarray into Original array

    for loop_var in range(left, right + 1):

        arr[loop_var] = temp_arr[loop_var]



    return inv_count




arr = [1, 20, 6, 4, 5]

n = len(arr)

result = mergeSort(arr, n)

print("Number of inversions are", result)
```

```
Number of inversions are 5
```