Merge Sort: Counting Inversions

In an array, arr, the elements at indices i and j (where $i < j$) form an inversion if $arr[i] > arr[j]$. In other words, inverted elements $arr[i]$ and $arr[j]$ are considered to be "out of order". To correct an inversion, we can swap adjacent elements.

For example, consider the dataset $arr = [2, 4, 1]$. It has two inversions: $(4, 1)$ and $(2, 1)$. To sort the array, we must perform the following two swaps to correct the inversions:

$$arr = [2, 4, 1] \xrightarrow{swap(arr[1], arr[2]) \rightarrow swap(arr[0], arr[1])} [1, 2, 4]$$

Given d datasets, print the number of inversions that must be swapped to sort each dataset on a new line.

Function Description

Complete the function *countInversions* in the editor below. It must return an integer representing the number of inversions required to sort the array.

countInversions has the following parameter(s):

- *arr*: an array of integers to sort .

Input Format

The first line contains an integer, d, the number of datasets.

Each of the next d pairs of lines is as follows:

1. The first line contains an integer, n, the number of elements in arr.
2. The second line contains n space-separated integers, $arr[i]$.

Constraints

- $1 \leq d \leq 15$
- $1 \leq n \leq 10^5$
- $1 \leq arr[i] \leq 10^7$

Output Format

For each of the d datasets, return the number of inversions that must be swapped to sort the dataset.

Sample Input

```
2
5
1 1 1 2 2
5
2 1 3 1 2
```

Sample Output

```
0
4
```

Explanation

We sort the following $d = 2$ datasets:

1. $arr = [1, 1, 1, 2, 2]$ is already sorted, so there are no inversions for us to correct. Thus, we print 0 on a new line.
2. $arr = [2, 1, 3, 1, 2] \xrightarrow{\text{1 swap}} [1, 2, 3, 1, 2] \xrightarrow{\text{2 swaps}} [1, 1, 2, 3, 2] \xrightarrow{\text{1 swap}} [1, 1, 2, 2, 3]$

We performed a total of $1 + 2 + 1 = 4$ swaps to correct inversions.