Merge Sort: Counting Inversions

In an array, $arr = [arr_0, arr_1, \ldots, arr_{n-1}]$, 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 $arr = [2, 4, 1]$. It has two inversions: $(2, 1)$ and $(4, 1)$. To sort the array, we must perform the following two swaps to correct the inversions:

$$arr = [2,4,1] \xrightarrow{swap(arr_1, arr_2) \to swap(arr_0, arr_1)} [1,2,4]$$

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

Input Format

The first line contains an integer, d, denoting the number of datasets.
The $2d$ subsequent lines describe each respective dataset over two lines:

1. The first line contains an integer, n, denoting the number of elements in the dataset.
2. The second line contains n space-separated integers describing the respective values of $arr_0, arr_1, \ldots, arr_{n-1}$.

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

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. $arr = [2,1,3,1,2] \xrightarrow{\text{1 swap}} [1,2,3,1,2] \xrightarrow{\text{2 swaps}} [1,1,2,3,2] \xrightarrow{\text{1 swap}} [1,1,2,2,3]$

As we performed a total of $1 + 2 + 1 = 4$ swaps to correct inversions, we print 4 on a new line.