

Merge Sort: Counting Inversions ★

Problem

Submissions

Leaderboard

Editorial

RATE THIS CHALLENGE



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.

Example

arr = [2, 4, 1]

To sort the array, we must perform the following two swaps to correct the inversions:

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

The sort has two inversions: (4, 1) and (2, 1).

Given an array *arr*, return the number of inversions to sort the array.

Function Description

Complete the function countInversions in the editor below.

countInversions has the following parameter(s):

- int arr[n]: an array of integers to sort

Returns

- int: the number of inversions

Input Format

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

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

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

Constraints

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

Sample Input

STDIN	Function
-----	-----
2	d = 2
5	arr[] size n = 5 for the first dataset
1 1 1 2 2	arr = [1, 1, 1, 2, 2]
5	arr[] size n = 5 for the second dataset
2 1 3 1 2	arr = [2, 1, 3, 1, 2]

Sample Output

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

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

[Change Theme](#)

Language

C#



```
24     */
25
26     public static long countInversions(List<int> arr)
27     {
28         return MergeSort(arr.ToArray());
29     }
30
31     private static long MergeSort(Span<int> arr)
32     {
33         long swaps = 0;
34         var center = arr.Length / 2;
35         if(arr.Length > 1)
36         {
37             swaps += MergeSort(arr.Slice(0, center));
38             swaps += MergeSort(arr.Slice(center));
39             swaps += Merge(arr, center);
40         }
41         return swaps;
42     }
43
44     private static long Merge(Span<int> arr, int startOfRightHalf)
45     {
46         long swaps = 0;
47         var unsorted = arr.ToArray();
48         var lhs = 0;
49         var rhs = startOfRightHalf;
50         var offset = 0;
```

Line: 104 Col: 1

Upload Code as File

☐ Test against custom input

Run Code

Submit Code