Merge Sort: Counting Inversions ★

Problem Submissions Leaderboard Editorial

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:

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

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

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

```
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.

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.

Change Theme JavaScript (Node.js)

```
// Complete the countInversions function below.
function countInversions(arr) {
let count=0;
mergesort(arr)

function mergesort(myar){
if (myar.length==1) return myar;
let mid= Math.floor(myar.length/2);
const a1= mergesort(myar.slice(0,mid));
const a2= mergesort(myar.slice(mid));
return merge(a1,a2);
}

function merge(a1,a2){
let ptrA=0;
let ptrB=0;
const NewArr=[];
while ((ptrA <a1.length) && (ptrB < a2.length)){
if (a1[ptrA] <= a2[ptrB]){
NewArr.push(a1[ptrA]);
ptrA++
} else {
NewArr.push(a2[ptrB]);
ptrB++
// and hereis the extra line to get the count
```

Line: 38 Col: 2

Upload Code as File Test against custom input Run Code Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Sample Test case 2

Input (stdin) Download

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

Your Output (stdout)

```
0
4
```

Expected Output Download

```
0
```