Codility_

CodeCheck Report: training4QDRYJ-TFB

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
ArrayInversionCount ⚠ Python	4 min	100%

Total score

100%

Tasks Details

Medium		Task Score	Correctness	Performance
	1. ArrayInversionCount Compute number of inversion in an array.	100%	100%	100%

Task description

An array A consisting of N integers is given. An *inversion* is a pair of indexes (P, Q) such that P < Q and A[Q] < A[P].

Write a function:

```
def solution(A)
```

that computes the number of inversions in A, or returns −1 if it exceeds 1,000,000,000.

For example, in the following array:

```
A[0] = -1 A[1] = 6 A[2] = 3
A[3] =  4 A[4] = 7 A[5] = 4
```

there are four inversions:

```
(1,2)  (1,3)  (1,5)  (4,5)
```

so the function should return 4.

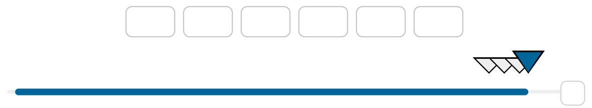
Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [0..100,000];
- each element of array A is an integer within

Solution

Programming language used:	Python
Total time used:	4 minutes ?
Effective time used:	4 minutes ?
Notes:	*not defined yet*

Task timeline ?



08:33:05 08:37:03

Code: 08:37:01 UTC, py, final, score: **100**

show code in pop-up

```
# you can write to stdout for debugging pur
```

the range [−2,147,483,648..2,147,483,647].

```
# print("this is a debug message")

def solution(A):
    # Implement your solution here
    def merge_and_count_inversions(left, ri
        merged = []
        inversions = 0
        i, j = 0, 0

        while i < len(left) and j < len(rig
            if left[i] <= right[j]:
                merged.append(left[i])
                i += 1
            else:
                merged.append(right[j])
                inversions += len(left) - i
                j += 1

        merged.extend(left[i:])
        merged.extend(right[j:])
        return merged, inversions


    def merge_sort_and_count_inversions(arr
        n = len(arr)
        if n <= 1:
            return arr, 0

        mid = n // 2
        left_half, left_count = merge_sort_
        right_half, right_count = merge_sor
        merged, merge_count = merge_and_cou

        total_count = left_count + right_co
        return merged, total_count


    _, inversions = merge_sort_and_count_in

    if inversions > 1000000000:
        return -1
    else:
        return inversions

```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N*log(N))**

expand all **Example tests**

▶ example1 ✔ OK
example test

expand all **Correctness tests**

▶ simple1 ✔ OK

▶ simple2 ✔ OK

▶ simple3 ✔ OK

▶ extreme_0_inv ✔ OK
[0], [], [1,2,3], [1,1,1]

▶ medium1 ✔ OK
n=100

▶ medium2 n=200	✔ OK

expand all

Performance tests

▶ medium3 n=1000	✔ OK
▶ big1 n=10000	✔ OK
▶ big2 n=20000	✔ OK
▶ big3 n=30000	✔ OK
▶ big_monotonic long descending and non-ascending sequence	✔ OK