Codility_

CodeCheck Report: trainingHM94EN-HNA

Test Name:

Check out Codility training tasks

Summary   Timeline   AI Assistant Transcript

Tasks summary

Task	Time spent	Score
ArrayInversionCount ⚠ C#	2 min	63%

Total score

63%

Tasks Details

Medium

1. ArrayInversionCount
Compute number of inversion in an array.

Task Score	Correctness	Performance
63%	100%	20%

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:

```
class Solution { public int solution(int[] 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:

Solution

Programming language used:	C#
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	*not defined yet*

Task timeline

02:03:02   02:04:32

- N is an integer within the range [0..100,000];
- each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

Code: 02:04:32 UTC, cs, final, score: **63**

show code in pop-up

```
using System;
// you can also use other imports, for example:
// using System.Collections.Generic;

// you can write to stdout for debugging purposes,
// Console.WriteLine("this is a debug message");

class Solution {
    public int solution(int[] A) {
        // Implement your solution here
            int inversionsCount = 0;

    for (int i = 0; i < A.Length; i++)
    {
        for (int j = i+1; j < A.Length; j++)
        {
            if (A[i] > A[j])
            {
                inversionsCount++;
            }
        }
    }

    return inversionsCount;
    }
}
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

Detected time complexity: **O(N**2)**

expand all	Example tests
▶ example1 example test	✓ OK
expand all	**Correctness tests**
▶ simple1	✓ OK
▶ simple2	✓ OK
▶ simple3	✓ OK
▶ extreme_0_inv [0], [], [1,2,3], [1,1,1]	✓ OK
▶ medium1 n=100	✓ OK
▶ medium2 n=200	✓ OK
expand all	**Performance tests**
▶ medium3 n=1000	✓ OK
▶	

	big1 n=10000	✗ TIMEOUT ERROR running time: 0.188 sec., time limit: 0.100 sec.
▶	big2 n=20000	✗ TIMEOUT ERROR running time: 0.732 sec., time limit: 0.112 sec.
▶	big3 n=30000	✗ TIMEOUT ERROR running time: 1.724 sec., time limit: 0.128 sec.
▶	big_monotonic long descending and non-ascending sequence	✗ TIMEOUT ERROR running time: 1.332 sec., time limit: 0.144 sec.