Codility_

Candidate Report: trainingKS5YHT-H3Q

Check out Codility training tasks

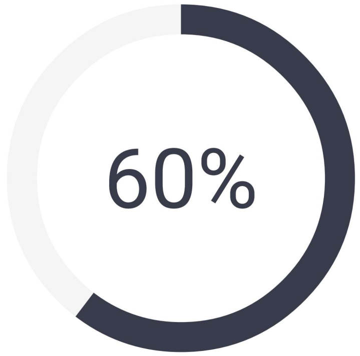
Test Name:

Summary Timeline Feedback

Tasks summary

Task	Time spent	Score
PassingCars C#	1 min	60%

Total score



Tasks Details

Easy

1. PassingCars

Count the number of passing cars on the road.

Task Score	Correctness	Performance
60%	100%	20%

Task description

A non-empty array A consisting of N integers is given. The consecutive elements of array A represent consecutive cars on a road.

Array A contains only 0s and/or 1s:

- 0 represents a car traveling east,
- 1 represents a car traveling west.

The goal is to count passing cars. We say that a pair of cars (P, Q), where 0 ≤ P < Q < N, is passing when P is traveling to the east and Q is traveling to the west.

For example, consider array A such that:

```
  A[0] = 0
  A[1] = 1
  A[2] = 0
  A[3] = 1
  A[4] = 1
```

We have five pairs of passing cars: (0, 1), (0, 3), (0, 4), (2, 3), (2, 4).

Write a function:

```
class Solution { public int solution(int[] A); }
```

Solution

Programming language used:	C#
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline

18:34:59 18:35:22

Code: 18:35:22 UTC, cs, final, score: **60**

show code in pop-up

that, given a non-empty array A of N integers, returns the number of pairs of passing cars.

The function should return −1 if the number of pairs of passing cars exceeds 1,000,000,000.

For example, given:

```
A[0] = 0
A[1] = 1
A[2] = 0
A[3] = 1
A[4] = 1
```

the function should return 5, as explained above.

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

- N is an integer within the range [1..100,000];
- each element of array A is an integer that can have one of the following values: 0, 1.

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

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

class Solution {
   public int solution(int[] A)
        {
            int retVal = 0;
            var len = A.Length;

            for (int i = 0; i < len; i++)
            {
                if (A[i] == 0)
                {
                    for (int j = i+1; j < len; j++)
                    {
                        retVal += A[j];

                    }
                }
            }

            return retVal;

        }
}
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

Detected time complexity: O(N ** 2)

expand all	Example tests	
▶ example example test		✓ OK
expand all	**Correctness tests**	
▶ single single element		✓ OK
▶ double two elements		✓ OK
▶ simple simple test		✓ OK
▶ small_random random, length = 100		✓ OK
▶ small_random2 random, length = 1000		✓ OK
expand all	**Performance tests**	
▶ medium_random random, length = ~10,000		✓ OK
▶ large_random random, length = ~100,000		✗ TIMEOUT ERROR running time: 2.248 sec., time limit: 0.144 sec.
▶ large_big_answer 0..01..1, length = ~100,000		✗ TIMEOUT ERROR running time: 3.364 sec., time

		limit: 0.144 sec.
▶	large_alternate 0101..01, length = ~100,000	✗ TIMEOUT ERROR running time: 2.252 sec., time limit: 0.144 sec.
▶	large_extreme large test with all 1s/0s, length = ~100,000	✗ TIMEOUT ERROR running time: 4.464 sec., time limit: 0.144 sec.

The PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.