Codility_

Candidate Report: trainingWFNMRR-H8J

Check out Codility training tasks

Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
PassingCars Java 8	2 min	100%

Total score

100%

Tasks Details

Easy

1. PassingCars

Count the number of passing cars on the road.

Task Score	Correctness	Performance
100%	100%	100%

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); }
```

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:

Solution

Programming language used:	Java 8
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	*not defined yet*

Task timeline

15:55:04 15:56:27

Code: 15:56:27 UTC, java, final, score: 100

show code in pop-up

```
// you can also use imports, for example:
// import java.util.*;

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

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

```
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.

```
        int countOne = 0, result = 0, n=A.length;
        while (n >= 1)
        {
            if (A[n-1] == 1)
                countOne++;
            else
                result += countOne;
            n--;

            if(result>1000000000)
                return -1;
        }
        return result;
    }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

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

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	✓ OK
▶	large_big_answer 0..01..1, length = ~100,000	✓ OK
▶	large_alternate 0101..01, length = ~100,000	✓ OK
▶	large_extreme large test with all 1s/0s, length = ~100,000	✓ OK

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.