Codility_

CodeCheck Report: training5QXZ3E-KGC

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task		Time spent	Score
PassingCars Python	⚠	14 min	100%

Total score

Tasks Details

		Task Score	Correctness	Performance
Easy	1. **PassingCars** Count the number of passing cars on the road.	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
```

Solution

Programming language used:	Python	
Total time used:	14 minutes	?
Effective time used:	14 minutes	?
Notes:	*not defined yet*	

Task timeline ?

```
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:

```
def solution(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:

```
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.

14:31:23 14:45:19

Code: 14:45:18 UTC, py, final, score: **100** show code in pop-up

```
# you can write to stdout for debugging
# print("this is a debug message")

def solution(A):
    # Implement your solution here
    # pass
    count = 0
    east_cars = 0

    for car in A:
        if car == 0:
            east_cars += 1
        else:
            count += east_cars

    return count if count <= 1_000_000_


```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all **Example tests**

▶ example ✔ OK
example test

expand all **Correctness tests**

▶ single ✔ OK
single element

▶ double ✔ OK
two elements

▶ simple ✔ OK
simple test

▶ small_random ✔ OK
random, length = 100

▶ small_random2 ✔ OK
random, length = 1000

expand all **Performance tests**

▶ medium_random ✔ OK
random, length = ~10,000

▶ large_random ✔ OK
random, length = ~100,000

▶ large_big_answer ✔ OK
0..01..1, length = ~100,000

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