codility

Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

100 out of 100 points

100%

Tasks in Test

	Time Spent	Task Score
PassingCars Submitted in: Python	4 min	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:

```
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:

Solution

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

Task timeline

13:32:04 13:35:22

Code: 13:35:22 UTC, py, final, score: **100**

show code in pop-up

```
1  from itertools import accumulate
2
3
4  def solution(a):
5      suffix_sum = list(accumulate(reversed(a)))
6      counter = 0
7      prev = suffix_sum[0]
8      for value in suffix_sum[1:]:
9
```

```
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.


```
10          if value == prev:
11              counter += prev
12          else:
13              prev = value
14          if counter > 1000000000:
15              return -1
        return counter
```

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