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: Scala	2 min	100%

TASKS DETAILS

1. PassingCars

EASY

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 $0 \le P < Q < N$, 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
```

Solution

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

Task timeline

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:

object Solution { def solution(a: Array[Int]): Int }

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.

07:11:08 07:12:31

Code: 07:12:31 UTC, scala, final, score: **100** show code in pop-up

```
1  import scala.collection.JavaConverters._
2
3  // you can write to stdout for debugging purposes, e.
4  // println("this is a debug message")
5
6  object Solution {
7    def solution(a: Array[Int]): Int = {
8    val seq = a.toSeq
9    val ret = seq.foldLeft((0, 0L)) {
10     case ((eastCount, passingCount), car) => {
11       car match {
12         case 0 => (eastCount + 1, passingCount)
13         case 1 => (eastCount, passingCount + eastCou
14       }
15     }
16   }
17
18   ret._2 match {
19     case x if x > 1000000000 => -1
20     case x                   => x.toInt
21   }
22  }
23 }
```

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