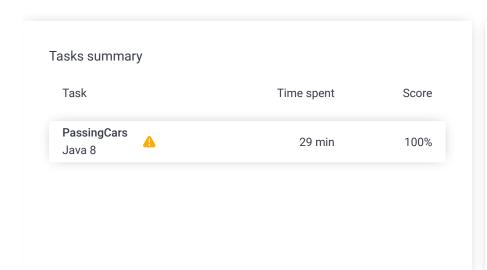
# Codility\_

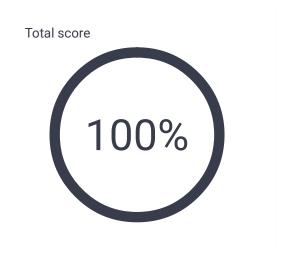
## CodeCheck Report: trainingBX8XAA-CXV

Test Name:

Summary Timeline

Check out Codility training tasks





#### **Tasks Details**

1. PassingCars
Count the number of passing cars on the road.

Task Score

Correctness

100%

Performance

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
- 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:

#### Solution

Programming language used: Java 8

Total time used: 29 minutes

Effective time used: 29 minutes

Notes: not defined yet

Task timeline

13:12:56 13:41:09

Code: 13:41:09 UTC, java,

show code in pop-up

final, score: 100

```
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:

```
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.

Copyright 2009–2021 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

#### Test results - Codility

```
// you can also use imports, for example:
 2
     // import java.util.*;
 3
     // you can write to stdout for debugging purposes,
 4
 5
     // System.out.println("this is a debug message");
 6
 7
     class Solution {
8
         public int solution(int[] A) {
9
             // write your code in Java SE 8
             final int MAX_RESULT = 10000000000;
10
              int result = 0, counter=0;
11
12
              for(int i=A.length-1; i>=0; i--) {
13
                  if(A[i]==1) {
                      counter++;
14
15
                  }
16
                  else {
                      if(result>MAX_RESULT) {
17
18
                          return -1;
19
                      }
20
                      result+=counter;
21
22
23
              return result;
24
         }
25
26
     }
```

## Analysis summary

The solution obtained perfect score.

#### **Analysis**

## Detected time complexity: O(N)

expand all		Example tests
•	example example test	✓ OK
expand all Corre		prrectness tests
•	single single element	✓ OK
<b>&gt;</b>	double two elements	√ OK
<b>&gt;</b>	simple simple test	√ OK
<b>&gt;</b>	small_random random, length = 100	√ OK
•	small_random2 random, length = 1000	✓ OK
ехра	nd all Pe	rformance tests
•	medium_random random, length = ~10,00	<b>✓ OK</b>
•	large_random random, length = ~100,0	<b>✓ 0K</b>
<b>•</b>	large_big_answer 0011, length = ~100,0	<b>✓ OK</b>

### Test results - Codility

•	large_alternate 010101, length = ~100,000	√ OK
•	large_extreme	✓ OK
	large test with all 1s/0s, length =	
	~100,000	