

CodeCheck Report: trainingBX8XAA-CXV

Test Name:

Check out Codility training tasks

Summary

Timeline

Tasks summary

Task	Time spent	Score
PassingCars Java 8	29 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 $0 \leq P < Q < N$, 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, final, score: 100

show code in pop-up

```
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

```
1 // you can also use imports, for example:
2 // import java.util.*;
3
4 // you can write to stdout for debugging purposes,
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
10        final int MAX_RESULT = 1000000000;
11        int result = 0, counter=0;
12        for(int i=A.length-1; i>=0; i--) {
13            if(A[i]==1) {
14                counter++;
15            }
16            else {
17                if(result>MAX_RESULT) {
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)**

Example tests	
▶ example	✓ OK
example test	
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	
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	

Test results - Codility

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