

Screen Report: Anonymous

Test Name:

[Check out Codility training tasks](#)

Summary Timeline

Tasks summary

| Task | Effective time spent | Score |
|--------------------|----------------------|-------|
| PassingCars C++ | 14 min | 100% |

Total score



Tasks Details

| | | | | | |
|------|---|------------|-------------|-------------|------|
| Easy | 1. PassingCars | Task Score | Correctness | Performance | |
| | 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 $0 \leq P < Q < N$, 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:

```
int solution(vector<int> &A);
```

that, given a non-empty array A of N integers, returns the number of pairs of passing cars.

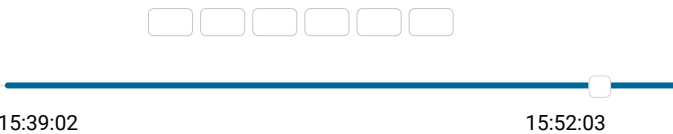
Solution

Programming language used: C++

Time spent on task: 14 minutes

Notes: not defined yet

Task timeline



Code: 15:52:03 UTC, cpp, final, [show code in pop-up](#)
score: 100

```
1 // you can use includes, for example:
2 // #include <algorithm>
3
4 // you can write to stdout for debugging purposes, e.g.
5 // cout << "this is a debug message" << endl;
6
7 int solution(vector<int> &A) {
```

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 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
8 // Implement your solution here
9
10 // create a suffix sum vector in which each element e
11 vector<int> suffixSum(A.size()+1 , 0);
12
13 for(int i = A.size()-1; i >= 0; i--)
14 {
15     suffixSum[i] = suffixSum[i+1] + A[i];
16 }
17
18 long long int numOfPassingCars{0};
19 for(int i = 0; i < A.size(); i++)
20 {
21     if(A[i] == 0)
22     {
23         numOfPassingCars += suffixSum[i];
24     }
25 }
26
27 if(numOfPassingCars > 1000000000)
28 {
29     return -1;
30 }
31
32 return numOfPassingCars;
33 }
```

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