Codility_

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

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:

```
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

15:39:02 15:52:03

Code: 15:52:03 UTC, cpp, final, score: **100**

show code in pop-up

```
// you can use includes, for example:
// #include <algorithm>

// you can write to stdout for debugging purposes, e.g.
// cout << "this is a debug message" << endl;

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.

```
    // Implement your solution here

    // create a suffix sum vector in which each element
    vector<int> suffixSum(A.size()+1 , 0);

    for(int i = A.size()-1; i >= 0; i--)
    {
        suffixSum[i] = suffixSum[i+1] + A[i];
    }

    long long int numOfPassingCars{0};
    for(int i = 0; i < A.size(); i++)
    {
        if(A[i] == 0)
        {
            numOfPassingCars += suffixSum[i];
        }
    }

    if(numOfPassingCars > 1000000000)
    {
        return -1;
    }

    return  numOfPassingCars;
}
```

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