Codility_

## Candidate Report: training9NDQTC-UCH

Check out Codility training tasks

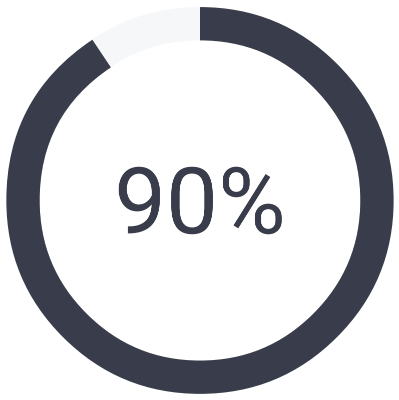
Test Name:

Summary Timeline

### Tasks summary

| Task | Time spent | Score |
|---|---|---|
| **PassingCars** C | 26 min | 90% |

### Total score


90%

## Tasks Details

Easy

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

| Task Score | Correctness | Performance |
|---|---|---|
| 90% | 100% | 80% |

### 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.

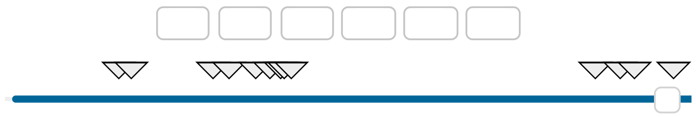
For example, consider array A such that:

```
A[0] = 0
A[1] = 1
A[2] = 0
```

### Solution

| | |
|---|---|
| Programming language used: | C |
| Total time used: | 26 minutes |
| Effective time used: | 26 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:

```
int solution(int A[], int N);
```

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.

09:45:21 10:11:21

Code: 10:11:21 UTC, c, final, score: **90**

show code in pop-up

```
// you can write to stdout for debugging purposes, e
// printf("this is a debug message\n");
int *prefix_sums(int *A, int N){
    int *P = (int *)malloc((N+1)*sizeof(int));
    P[0] = 0;
    for (int i = 1; i<(N+1); i++){
        P[i] = P[i-1] + A[i-1];
        //printf("%d ", P[i]);
    }
    return P;
}

int solution(int A[], int N) {
    // write your code in C99 (gcc 6.2.0)
    int ret = 0;
    int *P = prefix_sums(A,N);

    for (int i = 0; i<N; i++){
        if (!A[i]) {
            ret += P[N]-P[i];
            //printf("[%d i%d]", P[N], i);
        }
    }

    free(P);
    return ret<=1000000000? ret:-1;
}
```

## Analysis summary

The following issues have been detected: wrong answers.

## Analysis

Detected time complexity: **O(N)**

| expand all | Example tests | |
|---|---|---|
| ▶ | example<br>example test | ✓ OK |
| expand all | Correctness tests | |
| ▶ | single<br>single element | ✓ OK |
| ▶ | double<br>two elements | ✓ OK |
| ▶ | simple<br>simple test | ✓ OK |
| ▶ | small_random<br>random, length = 100 | ✓ OK |

| | | |
|---|---|---|
| ▶ | small_random2<br>random, length = 1000 | ✓ OK |
| expand all | Performance tests | |
| ▶ | medium_random<br>random, length = ~10,000 | ✓ OK |
| ▶ | large_random<br>random, length = ~100,000 | ✓ OK |
| ▼ | large_big_answer<br>0..01..1, length = ~100,000 | ✗ **WRONG ANSWER**<br>got -1794967296<br>expected -1 |
| | 1. 0.004 s **WRONG ANSWER**, got -1794967296 expected -1<br>2. 0.001 s **OK** | |
| ▶ | large_alternate<br>0101..01, length = ~100,000 | ✓ OK |
| ▶ | large_extreme<br>large test with all 1s/0s, length = ~100,000 | ✓ OK |