codility

Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary | Timeline

Test Score

90 out of 100 points

90%

Tasks in Test

	Time Spent	Task Score
PassingCars Submitted in: Java	2 min	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
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:

```
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.

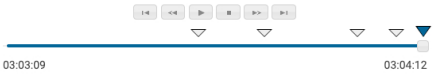
Write an **efficient** algorithm for the following assumptions:

- 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.

Solution

Programming language used:	Java
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	*not defined yet*

Task timeline



03:03:09 03:04:12

Code: 03:04:12 UTC, java, final, score: **90** show code in pop-up

```
1  import java.util.*;
2
3  class Solution {
4      public int solution(int[] A) {
5          // Max heap
6          PriorityQueue<Integer> pQueue = new PriorityQueue<>(Collectio
7
8          for (int i = 0; i < A.length; i++) {
9              if (A[i] == 0) {
10                 pQueue.add(i);
11             }
12         }
13
14         int sum = 0;
15         Iterator itr = pQueue.iterator();
16         int lastInd = A.length - 1;
17         int passCount = 0;
18         while (itr.hasNext()) {
19             sum += lastInd - (int) itr.next() - passCount;
20
21             if (sum > 1000000000) {
22                 return -1;
23             }
24
25             passCount++;
26         }
27
28         return sum;
29     }
30 }
```

Analysis summary

The following issues have been detected: timeout errors.

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	✘ TIMEOUT ERROR running time: 0.956 sec., time limit: 0.672 sec.