Codility_

CodeCheck Report: trainingUBXV7W-FS6

Test Name:

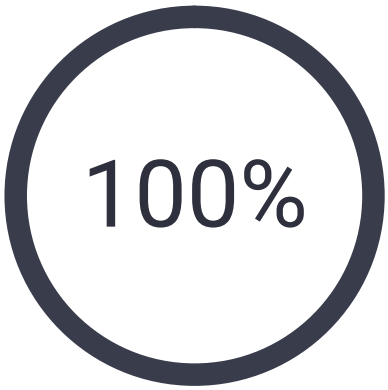
Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
EquiLeader Python	15 min	100%

Total score



Tasks Details

1. **EquiLeader**

Find the index S such that the leaders of the sequences A[0], A[1], ..., A[S] and A[S + 1], A[S + 2], ..., A[N - 1] are the same.

Easy

Task Score	Correctness	Performance
100%	100%	100%

Task description

A non-empty array A consisting of N integers is given.

The *leader* of this array is the value that occurs in more than half of the elements of A.

An *equi leader* is an index S such that 0 ≤ S < N − 1 and two sequences A[0], A[1], ..., A[S] and A[S + 1], A[S + 2], ..., A[N − 1] have leaders of the same value.

Solution

Programming language used:	Python
Total time used:	15 minutes
Effective time used:	15 minutes

For example, given array A such that:

```
A[0] = 4
A[1] = 3
A[2] = 4
A[3] = 4
A[4] = 4
A[5] = 2
```

we can find two equi leaders:

- 0, because sequences: (4) and (3, 4, 4, 4, 2) have the same leader, whose value is 4.
- 2, because sequences: (4, 3, 4) and (4, 4, 2) have the same leader, whose value is 4.

The goal is to count the number of equi leaders.

Write a function:

```
def solution(A)
```

that, given a non-empty array A consisting of N integers, returns the number of equi leaders.

For example, given:

```
A[0] = 4
A[1] = 3
A[2] = 4
A[3] = 4
A[4] = 4
A[5] = 2
```

the function should return 2, 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 within the range [−1,000,000,000..1,000,000,000].

Notes: *not defined yet*

Task timeline

11:55:09 12:09:43

Code: 12:09:43 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debuggin
2  # print("this is a debug message")
3
4  def solution(A):
5      # Implement your solution here
6      pass
7      stack = []
8      for num in A:
9          if not stack or stack[-1] == nu
10             stack.append(num)
11         else:
12             stack.pop()
13
14     if not stack:
15         return 0
16
17     leader = stack[-1]
18     leader_count = A.count(leader)
19     if leader_count <= len(A) // 2:
20         return 0
21
22     equi_leaders = 0
23     leader_count_so_far = 0
24     for i in range(len(A)):
25         if A[i] == leader:
26             leader_count_so_far += 1
27         if leader_count_so_far > (i + 1
28             equi_leaders += 1
29
30     return equi_leaders
31
```

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 small random test with two values, length = ~100	✔ OK
▶	small random + 200 * [MIN_INT] + random ,length = ~300	✔ OK

expand all	Performance tests	
▶	large_random large random test with two values, length = ~50,000	✔ OK
▶	large random(0,1) + 50000 * [0] + random(0, 1), length = ~100,000	✔ OK
▶	large_range 1, 2, ..., N, length = ~100,000	✔ OK
▶	extreme_large all the same values	✔ OK