

Summary

Timeline

Tasks summary

Task	Time spent	Score
EquiLeader Python	26 min	100%

Total score

100%

Tasks Details

Easy

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.

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 $0 \leq S < N - 1$ 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.

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, 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].

Copyright 2009–2022 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Solution

Programming language used: Python

Total time used:

26 minutes

?

Effective time used:

26 minutes

?

Notes:

not defined yet

Task timeline

21:15:05

21:40:36

Code: 21:40:35 UTC, py, final, score: 100

[show code in pop-up](#)

```
1  # you can write to stdout for debugging purposes, e.g.
2  # print("this is a debug message")
3
4  def solution(A):
5      # write your code in Python 3.6
6      if not A:
7          return 0
8      if len(A) == 1:
9          return 0
10     if len(A) == 2:
11         if A[0] == A[1]:
12             return 1
13         else:
14             return 0
15
16     dominator1 = {}
17     leader1 = None
18     max_arr = []
19     max_ = -1
20     for el in A:
21         if el not in dominator1.keys():
22             dominator1[el] = 1
23             max_ = max(1, max_)
24             max_arr.append(max_)
25         else:
26             dominator1[el] += 1
27             max_ = max(max_, dominator1[el])
28             max_arr.append(max_)
29     if max_ > len(A)/2:
30         leader1 = max(dominator1, key=dominator1.get)
31
32     count = 0
33     arr = []
34     dominator2 = {}
35     max2 = -1000000000-1
36     leader1 = None
37     leader2 = None
38     key = None
39     i = len(A)-2
40     while True:
41         el = A.pop()
42         if not A:
43             break
44         dominator1[el] -= 1
45         if max_arr[i] > len(A)/2:
46             leader1 = max(dominator1, key=dominator1.get)
47         else:
48             leader1 = None
49
50         arr.append(el)
51         if el not in dominator2.keys():
52             dominator2[el] = 1
53             tmp = max(max2, dominator2[el])
54             if tmp > max2:
55                 max2 = tmp
56                 key = el
57         else:
58             dominator2[el] += 1
59             tmp = max(max2, dominator2[el])
60             if tmp > max2:
61                 max2 = tmp
62                 key = el
63
64         if leader2 is None:
65             if dominator2[el] > len(arr)/2:
66                 leader2 = el
67         else:
68             if dominator2[leader2] > len(arr)/2:
69                 pass
70             else:
71                 if max2 > len(arr)/2:
72                     leader2 = key
73                     if key is None:
74                         leader2 = None
75                 else:
76                     leader2 = None
77
78         if leader1 is not None:
79             if leader2 is not None:
80                 if leader1 == leader2:
81                     count += 1
82
83         i -= 1
84
85     return count
```

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