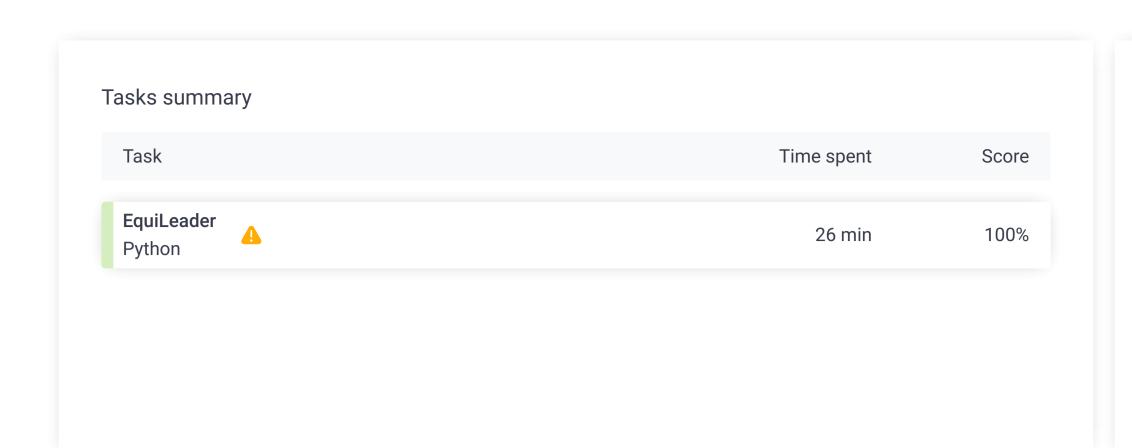
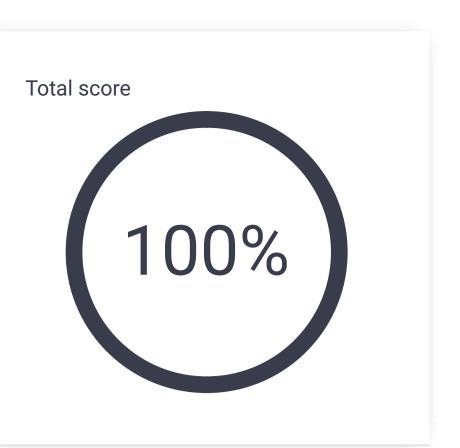
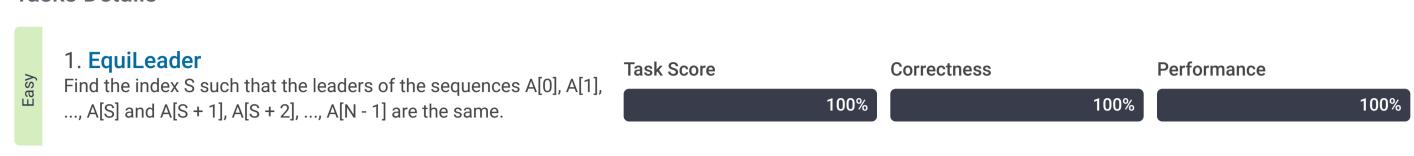
Summary Timeline





Tasks Details



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 $0 \le S < N - 1$ 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] = 3A[2] = 4

A[3] = 4

A[4] = 4A[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] = 4A[4] = 4

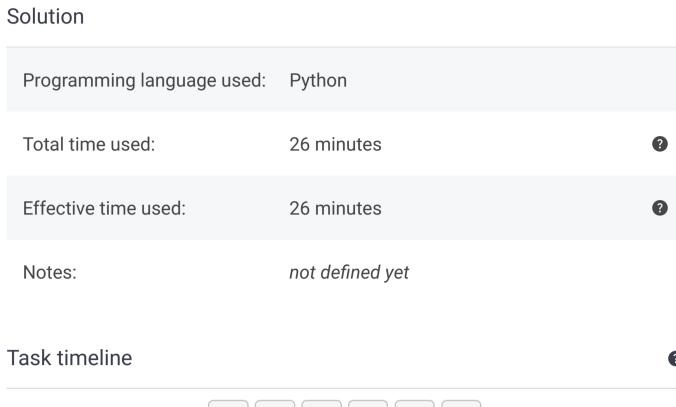
A[5] = 2 the function should return 2, as explained above.

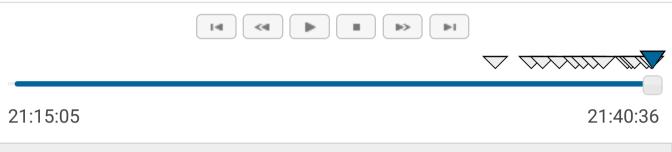
Write an **efficient** algorithm for the following assumptions:

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.





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

100

show code in pop-up

```
# you can write to stdout for debugging purposes, e.g.
    # print("this is a debug message")
    def solution(A):
4
        # write your code in Python 3.6
         if not A:
             return 0
         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
         dominator1 = {}
16
17
         leader1 = None
18
         max_arr = []
        \max_{} = -1
19
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])
                 max_arr.append(max_)
28
        if max_ > len(A)/2:
29
30
             leader1 = max(dominator1, key=dominator1.get)
31
32
         count = 0
33
         arr = []
34
         dominator2 = {}
35
         \max 2 = -10000000000-1
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
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
                     key = el
62
63
64
             if leader2 is None:
65
                 if dominator2[el] > len(arr)/2:
66
                     leader2 = el
67
             else:
                 if dominator2[leader2] > len(arr)/2:
68
69
                     pass
70
                 else:
71
                     if \max 2 > \frac{\text{len}(\text{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

