Codility_

CodeCheck Report: training5NDMHJ-P23

Test Name:

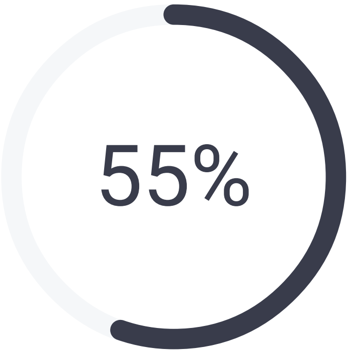
Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
EquiLeader C++ ⚠	33 min	55%

Total score



Tasks Details

1. EquiLeader

Easy

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
55%	100%	0%

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.

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:

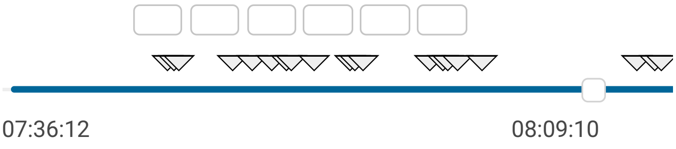
- 0, because sequences: (4) and (3, 4, 4, 4, 2) have the same leader, whose value is 4.
- 2, because sequences: (4, 3, 4) and (4, 4, 2) have the same leader, whose value is 4.

Solution

Programming language used:	C++	
Total time used:	33 minutes	?
Effective time used:	33 minutes	?
Notes:	*not defined yet*	

Task timeline

?



Code: 08:09:10 UTC, cpp, final, score: **55**

show code in pop-up

The goal is to count the number of equi leaders.

Write a function:

```
int solution(vector<int> &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].

```
// you can use includes, for example:
// #include <algorithm>
#include <climits>

// you can write to stdout for debugging purp
// cout << "this is a debug message" << endl;

int findLeader(vector<int>& A) {
    int leader = INT_MAX;

    int size = 0;
    int c;
    for (int a : A) {
        if (size == 0) {
            c = a;
            size = 1;
        }
        else {
            if (c == a)
                size++;
            else
                size--;
        }
    }

    int count = 0;

    for (int i = 0; i < A.size(); i++) {
        if (A[i] == c) {
            count++;
        }
    }
    if (count > A.size()/2)
        return c;
    return leader;
}

int solution(vector<int> &A) {
    // The leader of this array is the value 
    //n을 기준으로 0~n / n+1~end 으로 나눈 후에도 le
    int leader = findLeader(A);

    int count = 0;
    for (int i = 0; i < A.size(); i++) {
        vector<int> v1(A.begin(), A.begin()+i
        vector<int> v2(A.begin()+i+1, A.end()
        // for (auto e: v1) {cout << e << " "
        // cout << endl;
        // for (auto e: v2) {cout << e << " "
        // cout << endl;
        // int v1L = findLeader(v1);
        // int v2L = findLeader(v2);
        // cout << "leaders:" << endl;
        // cout << v1L <<" "<< v2L << " " << 
        if (findLeader(v1) != INT_MAX && findL
            count++;
    }

return count;
}
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

Detected time complexity: O(N ** 2)

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	✘ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▶ large random(0,1) + 50000 * [0] + random(0, 1), length = ~100,000	✘ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▶ large_range 1, 2, ..., N, length = ~100,000	✘ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▶ extreme_large all the same values	✘ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.