

CodeCheck Report: trainingQCAF5T-G4G

Test Name:

[Check out Codility training tasks](#)

Summary

Timeline

AI Assistant Transcript

Tasks summary

Task	Time spent	Score
Dominator	2 min	66%

Total score

66%

Tasks Details

Easy	1. <b>Dominator</b>	Task Score	Correctness	Performance
	Find an index of an array such that its value occurs at more than half of indices in the array.			
		66%	50%	100%

Task description

An array A consisting of N integers is given. The *dominator* of array A is the value that occurs in more than half of the elements of A.

For example, consider array A such that

A[0] = 3    A[1] = 4    A[2] = 3

A[3] = 2    A[4] = 3    A[5] = -1

A[6] = 3    A[7] = 3

The dominator of A is 3 because it occurs in 5 out of 8 elements of A (namely in those with indices 0, 2, 4, 6 and 7) and 5 is more than a half of 8.

Write a function

```
int solution(vector<int> &A);
```

that, given an array A consisting of N integers, returns index of any element of array A in which the dominator of A occurs. The function should return -1 if array A does not have a dominator.

For example, given array A such that

A[0] = 3    A[1] = 4    A[2] = 3

A[3] = 2    A[4] = 3    A[5] = -1

Solution

Programming language used:	C++
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	not defined yet

Task timeline

07:14:56

07:16:09

Code: 07:16:09 UTC, cpp, final, score: 66

[show code in pop-up](#)

1    // you can use includes, for example:

2    #include <algorithm>

A[6] = 3      A[7] = 3

the function may return 0, 2, 4, 6 or 7, as explained above.

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

- N is an integer within the range [0..100,000];
- each element of array A is an integer within the range [-2,147,483,648..2,147,483,647].

Copyright 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Test results - Codility

```
3  #include <map>
4
5  // you can write to stdout for debugging purposes
6  // cout << "this is a debug message" << endl;
7
8  int solution(vector<int> &A) {
9      int answer = -1;
10
11     map<int, int> m;
12
13     for (int i = 0; i < A.size(); i++) {
14         if (m.find(A[i]) == m.end())
15             m.insert(make_pair(A[i], 1));
16         else
17             m[A[i]]++;
18     }
19
20     vector<pair<int, int>> v(m.begin(), m.end());
21
22     sort(v.begin(), v.end(), [](pair<int, int> a, pair<int, int> b) {
23
24         // for (auto e : v) {
25         //     cout << e.first << " " << e.second << endl;
26         // }
27
28         if (v[0].second != v[1].second){
29
30             int leader = v[0].first;
31
32             for (int i = 0; i < A.size(); i++) {
33                 if (A[i] == leader)
34                     return i;
35             }
36         }
37
38         return answer;
39     }
```

Analysis summary

The following issues have been detected: wrong answers, runtime errors.

For example, for the input [] the solution terminated unexpectedly.

Analysis

expand all		Example tests	
▶	example	✓	OK
example test			
expand all		Correctness tests	
▶	small_nondominator	✓	OK
all different and all the same elements			
▶	small_half_positions	✗	WRONG ANSWER
half elements the same, and half + 1 elements the same		got 10, but element is not a dominator, value 2 has only 10 occurrences (n=20)	
▶	small	✓	OK
small test			
▶	small_pyramid	✓	OK
decreasing and plateau, small			
▶	extreme_empty_and_single_items	✗	RUNTIME ERROR
empty and single element arrays		tested program	

terminated with exit code		
1		
▶ extreme_half1	array with exactly N/2 values 1, N even + [0,0,1,1,1]	✗ WRONG ANSWER got 1, but element is not a dominator, value 1 has only 2 occurrences (n=4)
▶ extreme_half2	array with exactly floor(N/2) values 1, N odd + [0,0,1,1,1]	✗ WRONG ANSWER got 1, but element is not a dominator, value 1 has only 2 occurrences (n=5)
▶ extreme_half3	array with exactly ceil(N/2) values 1 + [0,0,1,1,1]	✓ OK
expand all Performance tests		
▶ medium_pyramid	decreasing and plateau, medium	✓ OK
▶ large_pyramid	decreasing and plateau, large	✓ OK
▶ medium_random	random test with dominator, N = 10,000	✓ OK
▶ large_random	random test with dominator, N = 100,000	✓ OK