codility

Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

100 out of 100 points

100%

Tasks in Test

	Time Spent	Task Score
Dominator Submitted in: Scala	23 min	100%

TASKS DETAILS

1. **Dominator**
Find an index of an array such that its value occurs at more than half of indices in the array.

EASY

Task Score	Correctness	Performance
100%	100%	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

Solution

Programming language used:	Scala
Total time used:	23 minutes
Effective time used:	23 minutes
Notes:	*not defined yet*

Task timeline

object Solution { def solution(a: Array[Int]): Int }

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
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].

08:28:53 08:50:56

Code: 08:50:56 UTC, scala, final, score: **100** show code in pop-up

```
1  import scala.collection.JavaConverters._
2
3  // you can write to stdout for debugging purposes, e
4  // println("this is a debug message")
5
6
7  object Solution {
8    def solution(a: Array[Int]): Int = {
9      val ret = a.toSeq.foldLeft((Map.empty[Int, Int], 0)
10       case ((container, indexOfMax), x) =>
11         val count         = container.getOrElse(x, 0) + 1
12         val currentMaxCount = container.getOrElse(inde
13
14         val newContainer = container updated (x, count)
15         count > currentMaxCount match {
16           case true  => (newContainer, x)
17           case false => (newContainer, indexOfMax)
18         }
19     }
20
21     ret._1.getOrElse(ret._2, 0) > a.length / 2 match {
22       case true =>
23         a.zipWithIndex.find(t => t._1 == ret._2).map(_.
24           case Some(x) => x
25           case None    => -1
26         }
27       case false => -1
28     }
29   }
30 }
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N*log(N)) or O(N)**

expand all	Example tests	
▶ example example test		✔ OK
expand all	Correctness tests	

▶	small_nondominator all different and all the same elements	✔ OK
▶	small_half_positions half elements the same, and half + 1 elements the same	✔ OK
▶	small small test	✔ OK
▶	small_pyramid decreasing and plateau, small	✔ OK
▶	extreme_empty_and_single_item empty and single element arrays	✔ OK
▶	extreme_half1 array with exactly N/2 values 1, N even + [0,0,1,1,1]	✔ OK
▶	extreme_half2 array with exactly floor(N/2) values 1, N odd + [0,0,1,1,1]	✔ OK
▶	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