

CodeCheck Report: training6FDCHA-M8M

Test Name:

[Check out Codility training tasks](#)

SummaryTimeline

Tasks summary

Task	Time spent	Score
FirstUnique Java 8	1 min	100%

Total score

100%

Tasks Details

Easy	1. FirstUnique	Task Score	Correctness	Performance
	Find the first unique number in a given sequence.			
		100%	100%	100%

Task description

A non-empty array A consisting of N integers is given. The *unique number* is the number that occurs exactly once in array A.

For example, the following array A:

```
A[0] = 4
A[1] = 10
A[2] = 5
A[3] = 4
A[4] = 2
A[5] = 10
```

contains two unique numbers (5 and 2).

You should find the first unique number in A. In other words, find the unique number with the lowest position in A.

For above example, 5 is in second position (because A[2] = 5) and 2 is in fourth position (because A[4] = 2). So, the first unique number is 5.

Write a function:

```
class Solution { public int solution(int[] A); }
```

Solution

Programming language used:	Java 8
Total time used:	1 minutes?
Effective time used:	1 minutes?
Notes:	not defined yet

Task timeline

20:40:1720:40:58

Code: 20:40:57 UTC, java, final, score: 100

show code in pop-up

1// you can also use imports, for example:

2import java.util.*;

that, given a non-empty array A of N integers, returns the first unique number in A. The function should return -1 if there are no unique numbers in A.

For example, given:

A[0] = 1
A[1] = 4
A[2] = 3
A[3] = 3
A[4] = 1
A[5] = 2

the function should return 4. There are two unique numbers (4 and 2 occur exactly once). The first one is 4 in position 1 and the second one is 2 in position 5. The function should return 4 because it is unique number with the lowest position.

Given array A such that:

A[0] = 6
A[1] = 4
A[2] = 4
A[3] = 6

the function should return -1. There is no unique number in A (4 and 6 occur more than once).

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 [0..1,000,000,000].

Copyright 2009–2023 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
3
4 // you can write to stdout for debugging purposes
5 // System.out.println("this is a debug message");
6
7 class Solution {
8     public int solution(int[] A) {
9         Map<Integer, Integer> firstIndices = new HashMap<>();
10
11         for (int i = 0; i < A.length; i++) {
12             if (firstIndices.containsKey(A[i])) {
13                 continue;
14             } else {
15                 firstIndices.put(A[i], i);
16             }
17         }
18
19         int minIndex = Integer.MAX_VALUE;
20         for (int index : firstIndices.values()) {
21             minIndex = Integer.min(minIndex, index);
22         }
23
24         return minIndex == Integer.MAX_VALUE ? -1 : minIndex;
25     }
26 }
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity:

O(N * log(N))

expand all	Example tests
▶ example0	✓ OK
example	
▶ example1	✓ OK
example	
▶ example2	✓ OK
example	
expand all	Correctness tests
▶ extreme_single	✓ OK
single element	
▶ extreme_no_unique	✓ OK
no unique value and [1,2,3,4]	
▶ extreme_min_max_value	✓ OK
min/max values	
▶ small1	✓ OK
small correctness test	
▶ small2	✓ OK
small correctness test	
▶ small3	✓ OK
small correctness tests	
expand all	Performance tests
▶ medium1	✓ OK
medium tests with few unique values, N = 10,003,	

▶ medium2	✓ OK
medium tests with few unique values, N = 10,209,	
▶ large	✓ OK
large tests with many minimal and maximal values, N = 50,000	
▶ big1	✓ OK
large test, N = 100,000	
▶ big2	✓ OK
large test, N = 100,000	