Codility_

CodeCheck Report: trainingJ4PGC3-T7W

Test Name:

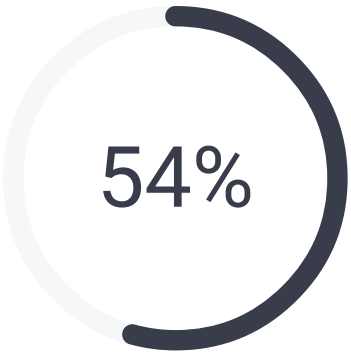
Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
FirstUnique Java 8 ⚠	22 min	54%

Total score



Tasks Details

Easy

1. FirstUnique

Find the first unique number in a given sequence.

Task Score	Correctness	Performance
54%	83%	20%

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:	22 minutes
Effective time used:	22 minutes
Notes:	*not defined yet*

Task timeline

20:03:05 20:24:24

Code: 20:24:24 UTC, java, final, score: **54** show code in pop-up

```
1  // you can also use imports, for example:
2  import java.util.*;
3
```

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 bacause 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].


```
4  // you can write to stdout for debugging purpo
5  // System.out.println("this is a debug message
6
7  class Solution {
8          public int solution(int[] A) {
9                  int min = Integer.MAX_VALUE;
10                 int max = -1;
11
12                 for (int i = 0; i < A.length;
13                         min = Integer.min(min,
14                         max = Integer.max(max,
15                 }
16
17                 int[] firstIndicies = new int
18                 for (int i = 0; i < A.length;
19                         int normalizedValue =
20                         if (firstIndicies[norm
21                                 firstIndicies
22                         } else {
23                                 firstIndicies
24                         }
25                 }
26
27                 int minIndex = Integer.MAX_VAL
28                 for (int normalizedValue = 0;
29                         if (firstIndicies[norm
30                                 minIndex = Mat
31                         }
32                 }
33
34                 return minIndex == Integer.MAX
35     }
36 }
```

Analysis summary

The following issues have been detected: runtime errors.

For example, for the input `[1000000000, 7, 1000000000]` the solution terminated unexpectedly.

Analysis

Detected time complexity: **O(N**2)**

collapse all	Example tests	
▼ example0 example		✔ OK
1. 0.008 s OK		
▼ example1 example		✔ OK
1. 0.004 s OK		
▼ example2 example		✔ OK
1. 0.008 s OK		
collapse all	Correctness tests	
▼ extreme_single single element		✔ OK
1. 0.004 s OK		
2. 0.004 s OK		

3. 0.004 s **OK**

▼ extreme_no_unique ✔ **OK**
no unique value and [1,2,3,4]

1. 0.004 s **OK**
2. 0.004 s **OK**

▼ extreme_min_max_value ✘ **RUNTIME ERROR**
min/max values
tested program terminated with exit code 1

1. 0.008 s **OK**
2. 0.028 s **RUNTIME ERROR**, tested program terminated with exit code 1

stderr:
```
Exception in thread "main" java.lang.OutOfMemo
        at Solution.solution(Solution.java:17)
        at Exec.run(exec.java:44)
        at Exec.main(exec.java:33)
```

▼ small1 ✔ **OK**
small correctness test

1. 0.008 s **OK**

▼ small2 ✔ **OK**
small correctness test

1. 0.004 s **OK**
2. 0.008 s **OK**

▼ small3 ✔ **OK**
small correctness tests

1. 0.008 s **OK**
2. 0.008 s **OK**

collapse all Performance tests

▼ medium1 ✔ **OK**
medium tests with few unique values, N = 10,003,

1. 0.028 s **OK**

▼ medium2 ✘ **RUNTIME ERROR**
medium tests with few unique values, N = 10,209,
tested program terminated with exit code 1

1. 0.068 s **RUNTIME ERROR**, tested program terminated with exit code 1

stderr:
```
Exception in thread "main" java.lang.OutOfMemo
        at Solution.solution(Solution.java:17)
        at Exec.run(exec.java:44)
        at Exec.main(exec.java:33)
```

▼ large ✘ **RUNTIME ERROR**
large tests with many minimal and maximal values, N = 50,000
tested program terminated with exit code 1

1. 0.200 s **RUNTIME ERROR**, tested program terminated with exit code 1

stderr:

```
Exception in thread "main" java.lang.OutOfMemo
        at Solution.solution(Solution.java:17)
        at Exec.run(exec.java:44)
        at Exec.main(exec.java:33)
```

2. 0.220 s **OK**

▼ big1 — large test, N = 100,000 — ✘ **RUNTIME ERROR** tested program terminated with exit code 1

1. 0.424 s **RUNTIME ERROR**, tested program terminated with exit code 1

stderr:

```
Exception in thread "main" java.lang.OutOfMemo
        at Solution.solution(Solution.java:17)
        at Exec.run(exec.java:44)
        at Exec.main(exec.java:33)
```

▼ big2 — large test, N = 100,000 — ✘ **RUNTIME ERROR** tested program terminated with exit code 1

1. 0.452 s **RUNTIME ERROR**, tested program terminated with exit code 1

stderr:

```
Exception in thread "main" java.lang.OutOfMemo
        at Solution.solution(Solution.java:17)
        at Exec.run(exec.java:44)
        at Exec.main(exec.java:33)
```