codility

Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

88 out of 100 points

88%

Tasks in Test

	Time Spent	Task Score
MissingInteger Submitted in: Java 8	17 min	88%

TASKS DETAILS

MEDIUM

1. **MissingInteger**
Find the smallest positive integer that does not occur in a given sequence.

Task Score	Correctness	Performance
88%	80%	100%

Task description

This is a demo task.

Write a function:

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

that, given an array A of N integers, returns the smallest positive integer (greater than 0) that does not occur in A.

For example, given A = [1, 3, 6, 4, 1, 2], the function should return 5.

Given A = [1, 2, 3], the function should return 4.

Given A = [−1, −3], the function should return 1.

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..1,000,000].

Solution

Programming language used:	Java 8
Total time used:	17 minutes
Effective time used:	17 minutes
Notes:	*not defined yet*

Task timeline

09:12:49 09:29:14

Code: 09:29:14 UTC, java, final, score: **88**

show code in pop-up

```
1  // you can also use imports, for example:
2  // import java.util.*;
3
4  // you can write to stdout for debugging purposes, e.g.
5  // System.out.println("this is a debug message");
6
7  class Solution {
8     public static int m = 1000000;
9          public static boolean[] mem;
```

```
10
11          public static int solution(int[] a) {
12
13                  mem = new boolean[m];
14                  for (int i = 0; i < a.length; i++) {
15                          if (a[i] > 0)
16                                  mem[a[i]] = true;
17                  }
18
19                  for (int i = 1; i < mem.length; i++) {
20                          if (!mem[i])
21                                  return i;
22                  }
23                  return 0;
24          }
25  }
```

Analysis summary

The following issues have been detected: runtime errors.

For example, for the input `[-1000000, 1000000]` the solution terminated unexpectedly.

Analysis

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

expand all — Example tests

▶ example1 first example test	✓ OK	
▶ example2 second example test	✓ OK	
▶ example3 third example test	✓ OK	

expand all — Correctness tests

▶ extreme_single a single element	✓ OK	
▶ simple simple test	✓ OK	
▼ extreme_min_max_value minimal and maximal values	✗ RUNTIME ERROR tested program terminated with exit code 1	

1. 0.004 s **RUNTIME ERROR**, tested program terminated with exit code 1

stderr:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBound
        at Solution.solution(Solution.java:16)
        at Exec.run(exec.java:48)
        at Exec.main(exec.java:34)
```

2. 0.004 s **RUNTIME ERROR**, tested program terminated with exit code 1

stderr:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBound
        at Solution.solution(Solution.java:16)
        at Exec.run(exec.java:48)
        at Exec.main(exec.java:34)
```

▶ positive_only shuffled sequence of 0...100 and then 102...200	✓ OK	

▶ negative_only shuffled sequence -100 ... -1	✓ OK

expand all Performance tests

▶ medium chaotic sequences length=10005 (with minus)	✓ OK
▶ large_1 chaotic + sequence 1, 2, ..., 40000 (without minus)	✓ OK
▶ large_2 shuffled sequence 1, 2, ..., 100000 (without minus)	✓ OK
▶ large_3 chaotic + many -1, 1, 2, 3 (with minus)	✓ OK