Codility_

Candidate Report: trainingUBG6TW-A85

Check out Codility training tasks

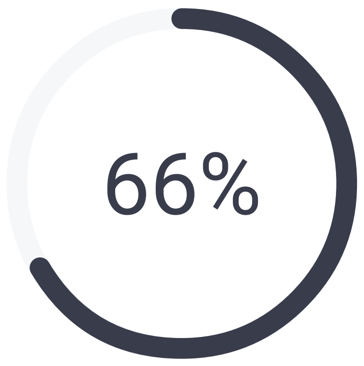
Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
MissingInteger ⚠ Java 8	79 min	66%

Total score



Identity verification
Add another layer of security with Identity Verification, learn more.

Tasks Details

Medium

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

Task Score	Correctness	Performance
66%	100%	25%

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.

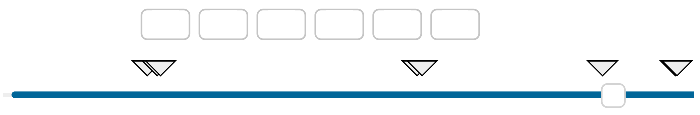
Write an **efficient** algorithm for the following assumptions:

- 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:	79 minutes
Effective time used:	79 minutes
Notes:	*not defined yet*

Task timeline



11:35:14 12:53:22

Code: 12:53:22 UTC, java, final, score: **66** show code in pop-up

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

// you can write to stdout for debugging purposes,
// System.out.println("this is a debug message");

class Solution {
    public int solution(int[] A) {
        int n,b=0;
        List<Integer> list1=new ArrayList<Integer>(
        for(int a:A){
            list1.add(a);
        }
        for(n=1;b==0;n++){
            if(!list1.contains(n)){
                return n;
            }
        }
        return n;

        // write your code in Java SE 8
    }
}
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

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

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	✓ OK
▸	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)	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
1.	6.000 s **TIMEOUT ERROR**, Killed. Hard limit reached: 6.000 sec.	
▸	large_2 shuffled sequence 1, 2, ..., 100000 (without minus)	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▸	large_3 chaotic + many -1, 1, 2, 3 (with minus)	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.