Codility_

## Candidate Report: trainingF9HN7G-D8R

Check out Codility training tasks

Test Name:

Summary Timeline Feedback

### Tasks summary

| Task | Time spent | Score |
|---|---|---|
| **MissingInteger** C# | 1 min | 100% |

### Total score


## Tasks Details

Medium

### 1. MissingInteger

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

| Task Score | Correctness | Performance |
|---|---|---|
| 100% | 100% | 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:

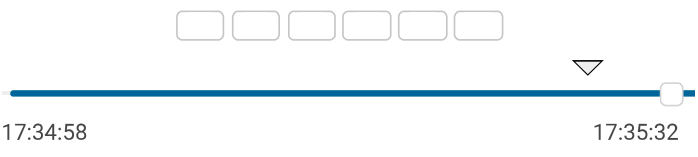
- N is an integer within the range [1..100,000];
- each element of array A is an integer within the range [−1,000,000..1,000,000].



### Solution

| | |
|---|---|
| Programming language used: | C# |
| Total time used: | 1 minutes |
| Effective time used: | 1 minutes |
| Notes: | *not defined yet* |

### Task timeline




Code: 17:35:32 UTC, cs, final, score: **100** show code in pop-up

```
using System;
using System.Linq;
// you can also use other imports, for example:
// using System.Collections.Generic;

// you can write to stdout for debugging purposes, e.g.
// Console.WriteLine("this is a debug message");

class Solution {
       bool isContiguous(int []arr)
        {
            int cnt = arr.Count();
            var max = arr.Max();
            var min = arr.Min();
            if (max - min == cnt - 1)
                return true;
            else
                return false; ;
        }

        public int solution(int[] A)
        {
            A = A.Where(x => x > 0).OrderBy(x => x).Distinc
            if (A.Length==0 || A.Count(x => x > 0) == 0 ||
            {
                return 1;
            }else
            {
                int len = A.Length;
                if (isContiguous(A))
                    return A[len-1]+1;
                else
                {
                    var max = A.Max();
                    var positiveSubset = A.Where(x => x >=
                    for (int i = 0; i < positiveSubset.Leng
                    {
                        if (i + 1 != positiveSubset[i])
                            return i + 1;
                    }
                }
            }
            return -1;
        }
}
```

## Analysis summary

The solution obtained perfect score.

## Analysis

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

| expand all | Example tests | |
|---|---|---|
| ▶ | example1<br>first example test | ✓ OK |
| ▶ | example2<br>second example test | ✓ OK |
| ▶ | example3<br>third example test | ✓ OK |
| | Correctness tests | |

expand all

| | | |
|---|---|---|
| ▶ | extreme_single<br>a single element | ✓ OK |
| ▶ | simple<br>simple test | ✓ OK |
| ▶ | extreme_min_max_value<br>minimal and maximal values | ✓ OK |
| ▶ | positive_only<br>shuffled sequence of 0...100 and then 102...200 | ✓ OK |
| ▶ | negative_only<br>shuffled sequence -100 ... -1 | ✓ OK |

expand all

## Performance tests

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