Codility_

Candidate Report: trainingDXVJPN-R5K

Check out Codility training tasks

Test Name:

Summary Timeline Feedback

Tasks summary

Task	Time spent	Score
PermMissingElem C#	1 min	100%

Total score

100%

Tasks Details

Easy

1. PermMissingElem
Find the missing element in a given permutation.

Task Score	Correctness	Performance
100%	100%	100%

Task description

An array A consisting of N different integers is given. The array contains integers in the range [1..(N + 1)], which means that exactly one element is missing.

Your goal is to find that missing element.

Write a function:

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

that, given an array A, returns the value of the missing element.

For example, given array A such that:

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

the function should return 4, as it is the missing element.

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

- N is an integer within the range [0..100,000];
- the elements of A are all distinct;

Solution

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

Task timeline

00:54:32 00:55:28

Code: 00:55:28 UTC, cs, final, score: **100**

show code in pop-up

- each element of array A is an integer within the range [1..(N + 1)].

```
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 {
            public int solution(int[] A)
        {
            if (A == null || A.Length == 0 )
                return 1;
            if (A.Length == 1)
            {
                if (A[0] == 1)
                    return 2;
                else
                    return 1;
            }

            A = A.OrderBy(x => x).ToArray();
            int ret = 0;
            var fullArray = Enumerable.Range(1, A.Max()).Tc
            if (fullArray.Length <= A.Length)
            {
                if (fullArray[0] == 1)
                    ret = fullArray[fullArray.Length - 1] +
                else
                    ret = 1;
            }else
                ret = fullArray.Except(A).First();
            return ret;
        }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example example test	✓ OK
expand all	Correctness tests	
▶	empty_and_single empty list and single element	✓ OK
▶	missing_first_or_last the first or the last element is missing	✓ OK
▶	single single element	✓ OK
▶	double two elements	✓ OK
▶	simple simple test	✓ OK

expand all	Performance tests
▶ medium1 medium test, length = ~10,000	✓ OK
▶ medium2 medium test, length = ~10,000	✓ OK
▶ large_range range sequence, length = ~100,000	✓ OK
▶ large1 large test, length = ~100,000	✓ OK
▶ large2 large test, length = ~100,000	✓ OK