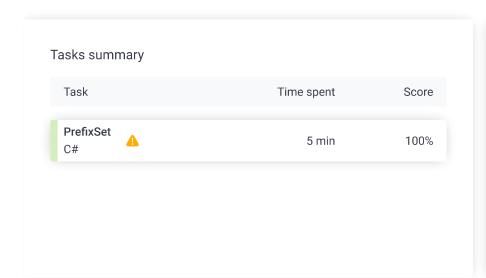
# Codility\_

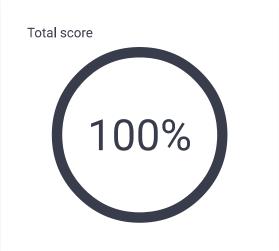
## CodeCheck Report: trainingB6DDE5-MEG

Test Name:

Timeline

Summary





Check out Codility training tasks

## Tasks Details

## 1. PrefixSet

Given a table A of N integers from 0 to N-1 calculate the smallest such index P, that that  $\{A[0],...,A[N-1]\} =$  $\{A[0],...,A[P]\}.$ 

Task Score Correctness Performance 100% 100% 100%

## Task description

A non-empty array A consisting of N integers is given. The first covering prefix of array A is the smallest integer P such that 0≤P<N and such that every value that occurs in array A also occurs in sequence A[0], A[1], ..., A[P].

For example, the first covering prefix of the following 5-element array A:

A[0] = 2

A[1] = 2

A[2] = 1

A[3] = 0

A[4] = 1

is 3, because sequence [A[0], A[1], A[2], A[3]] equal to [2, 2, 1, 0], contains all values that occur in array A.

## Solution

Programming language used:	C#		
Total time used:	5 minutes	•	
Effective time used:	5 minutes	•	
Notes:	not defined yet		
Task timeline	H >> 1	<b>3</b>	

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

that, given a non-empty array A consisting of N integers, returns the first covering prefix of A.

For example, given array A such that

```
A[0] = 2
A[1] = 2
A[2] = 1
A[3] = 0
A[4] = 1
```

the function should return 3, as explained above.

Write an efficient algorithm for the following assumptions:

- N is an integer within the range [1..1,000,000];
- each element of array A is an integer within the range [0..N-1].

Copyright 2009–2022 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
Code: 13:58:52 UTC, cs, final,
                                     show code in pop-up
score: 100
1
     using System;
2
     // you can also use other imports, for example:
3
     // using System.Collections.Generic;
4
5
     // you can write to stdout for debugging purposes,
6
     // Console.WriteLine("this is a debug message");
7
8
     class Solution {
         public int solution(int[] A) {
9
10
11
             var arr = new bool[A.Length];
12
             var res = 0;
13
             for (var i = 0; i < A.Length; i++) {</pre>
                  if (!arr[A[i]]) {
14
15
                      arr[A[i]] = true;
16
                     res = i;
17
18
             }
19
             return res;
20
         }
21
     }
```

## Analysis summary

The solution obtained perfect score.

## **Analysis**

expand all Example tests		
example1 example test	√ OK	
expand all	Correctness tests	
<ul><li>extreme_single</li><li>1-element sequence</li></ul>	√ OK	
<ul><li>extreme_two</li><li>1-element sequence</li></ul>	√ OK	
extreme_constar constant sequence	t ✓ OK	
extreme_identity identity permutation	√ OK	
extreme_permutation	ation ✓ <b>OK</b>	
simple1 very simple sequence	✓ OK	
► simple2 simple sequence	√ OK	

•	binary binary sequence	✓	OK
<b>•</b>	periodic	_/	OK
	periodic pattern	Ť	
expa	nd all Performance te	st	S
<b>&gt;</b>	random_dec_100 random test 100 elements, 37 different values.	✓	ОК
•	random_dec_1000 random test 1000 elements, 34 different values.	<b>√</b>	OK
•	random_dec_10000 random test 10 000 elements, 30 different values.	✓	OK
•	random_dec_100000 random test 100 000 elements, 27 different values.	✓	OK
•	random_sqrt_100 random test 100 elements, 10 different values.	<b>√</b>	OK
•	random_sqrt_1000 random test 1000 elements, 31 different values.	✓	OK
•	random_sqrt_10000 random test 10 000 elements, 100 different values.	✓	OK
•	random_sqrt_100000 random test 100 000 elements, 316 different values.	✓	OK
•	random_n_log_100 random test 100 elements and n/log_2 n values.	✓	OK
•	random_n_log_1000 random test 1000 elements and n/log_2 n values.	✓	OK
•	random_n_log_10000 random test 10 000 elements and n/log_2 n values.	✓	OK
•	random_n_log_100000 random test 100 000 elements and n/log_2 n values.	✓	OK
•	random_n_100 random test 100 elements and values.	✓	OK
•	random_n_1000 random test 1000 elements and values.	✓	OK
•	random_n_10000 random test 10 000 elements and values.	✓	OK
•	random_n_100000 random test 100 000 elements and values.	<b>√</b>	OK