

Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
PrefixSet C#	5 min	100%

Total score

100%

Tasks Details

Easy

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 $0 \leq P < N$ 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



Write a function

```
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.

13:54:32

13:58:53

Code: 13:58:52 UTC, cs, final,
score: 100

[show code in pop-up](#)

```
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 {
9      public int solution(int[] A) {
10
11          var arr = new bool[A.Length];
12          var res = 0;
13          for (var i = 0; i < A.Length; i++) {
14              if (!arr[A[i]]) {
15                  arr[A[i]] = true;
16                  res = i;
17              }
18          }
19          return res;
20      }
21  }
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **$O(N)$ or $O(N \cdot \log(N))$**

expand all

Example tests

▶ example1 ✓ OK
example test

expand all

Correctness tests

▶ extreme_single ✓ OK
1-element sequence

▶ extreme_two ✓ OK
1-element sequence

▶ extreme_constant ✓ OK
constant sequence

▶ extreme_identity ✓ OK
identity permutation

▶ extreme_permutation ✓ OK
permutation

▶ simple1 ✓ OK
very simple sequence

▶ simple2 ✓ OK
simple sequence

▶ binary	✓ OK
binary sequence	
▶ periodic	✓ OK
periodic pattern	
expand all	Performance tests
▶ random_dec_100	✓ OK
random test 100 elements, 37 different values.	
▶ random_dec_1000	✓ OK
random test 1000 elements, 34 different values.	
▶ random_dec_10000	✓ OK
random test 10 000 elements, 30 different values.	
▶ random_dec_100000	✓ OK
random test 100 000 elements, 27 different values.	
▶ random_sqrt_100	✓ OK
random test 100 elements, 10 different values.	
▶ random_sqrt_1000	✓ OK
random test 1000 elements, 31 different values.	
▶ random_sqrt_10000	✓ OK
random test 10 000 elements, 100 different values.	
▶ random_sqrt_100000	✓ OK
random test 100 000 elements, 316 different values.	
▶ random_n_log_100	✓ OK
random test 100 elements and $n/\log_2 n$ values.	
▶ random_n_log_1000	✓ OK
random test 1000 elements and $n/\log_2 n$ values.	
▶ random_n_log_10000	✓ OK
random test 10 000 elements and $n/\log_2 n$ values.	
▶ random_n_log_100000	✓ OK
random test 100 000 elements and $n/\log_2 n$ values.	
▶ random_n_100	✓ OK
random test 100 elements and values.	
▶ random_n_1000	✓ OK
random test 1000 elements and values.	
▶ random_n_10000	✓ OK
random test 10 000 elements and values.	
▶ random_n_100000	✓ OK
random test 100 000 elements and values.	