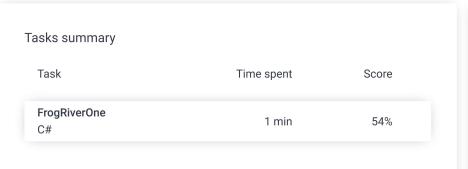
# Codility\_

# Candidate Report: training VPGJBJ-VCP

Check out Codility training tasks

Test Name:

Summary Timeline Feedback





Performance

0%

#### **Tasks Details**

asy

### 1. FrogRiverOne

Find the earliest time when a frog can jump to the other side of a river.

Task Score

54%

Correctness

#### Task description

A small frog wants to get to the other side of a river. The frog is initially located on one bank of the river (position 0) and wants to get to the opposite bank (position X+1). Leaves fall from a tree onto the surface of the river.

You are given an array A consisting of N integers representing the falling leaves. A[K] represents the position where one leaf falls at time K, measured in seconds.

The goal is to find the earliest time when the frog can jump to the other side of the river. The frog can cross only when leaves appear at every position across the river from 1 to X (that is, we want to find the earliest moment when all the positions from 1 to X are covered by leaves). You may assume that the speed of the current in the river is negligibly small, i.e. the leaves do not change their positions once they fall in the river.

For example, you are given integer X = 5 and array A such that:

- A[0] = 1
- A[1] = 3
- A[I] 2
- A[2] = 1
- A[3] = 4A[4] = 2
- A[5] = 3

### Solution

Task timeline

Programming language used: C#

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: not defined yet

21:50:44 21:51:35

Code: 21:51:34 UTC, cs, final, show code in pop-up score: 54

9/3/2020 Test results - Codility

```
A[6] = 5
A[7] = 4
```

In second 6, a leaf falls into position 5. This is the earliest time when leaves appear in every position across the river.

Write a function:

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

that, given a non-empty array A consisting of N integers and integer X, returns the earliest time when the frog can jump to the other side of the river.

If the frog is never able to jump to the other side of the river, the function should return -1.

For example, given X = 5 and array A such that:

```
A[\emptyset] = 1
A[1] = 3
A[2] = 1
A[3] = 4
A[4] = 2
A[5] = 3
A[6] = 5
A[7] = 4
```

the function should return 6, as explained above.

Write an efficient algorithm for the following assumptions:

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

Copyright 2009–2020 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
using System;
 2
     using System.Collections.Generic;
 3
     using System.Linq;
     // you can also use other imports, for example:
 4
 5
     // using System.Collections.Generic;
     // you can write to stdout for debugging purposes, e.g.
 8
     // Console.WriteLine("this is a debug message");
 9
10
     class Solution {
11
      public int solution(int X, int[] A)
12
              {
                  if (A.Length==1)
13
14
                      if (A[0] == X)
15
16
                      {
17
                           return 0;
18
                      }
19
                      else
20
                           return -1;
21
22
                  Dictionary<int,int> dict = new Dictionary<int,</pre>
23
                  for (int i = 1; i <= X; i++)</pre>
24
25
                      dict.Add(i, -1);
26
27
28
                  int second = 0;
29
                  do
30
31
                  {
                      if (dict.ContainsKey(A[second]))
32
                          dict[A[second]]++;
33
34
35
                      ++second;
36
                      if (second > A.Length - 1)
37
                          return -1;
38
                  } while (dict.Values.Count(x => x == -1) > 0);
39
40
                  return --second;
41
42
             }
43
     }
```

#### Analysis summary

The following issues have been detected: timeout errors.

## Analysis 2

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

expand all	Example tests	
example example te		
expand all	Correctness tests	
simple test	<b>√</b> 0K	
single elem	✓ OK nent	
extreme_ frog never	frog   OK  across the river	

<b>•</b>	small_r	andom1	<b>√</b>	ОК	
	3 random	permutation, X = 50			
•	small_random2 5 random permutation, X = 60			OK	
•		e_leaves in the same place	✓	OK	
collap	ose all	Performance t	ests		
•		n_random andom permutations, X = ~5,000	X	TIMEOUT ERROR running time: 0.608 sec., tim limit: 0.100 sec.	
1.	0.608 s TIMEOUT ERROR, running time: 0.608 sec., time limit: 0.100 sec				
2.	0.768 s TIMEOUT ERROR, running time: 0.768 sec., time limit: 0.100				
▼		n_range c sequences, X = 5,000	X	TIMEOUT ERROR running time: 0.420 sec., tim limit: 0.100 sec.	
1.	0.420 s	s TIMEOUT ERROR, running time: 0.420 sec., time limit: 0.100 sec			
•	large_ra	andom 00 random permutation, X =	X	TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.	
1.	6.000 s TIMEOUT ERROR, Killed. Hard limit reached: 6.000 sec.				
2.	0.140 s	ок			
•	large_p	ermutation ion tests	X	TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.	
1.	6.000 s	TIMEOUT ERROR, Killed. Hard	limit	reached: 6.000 sec.	
2.	6.000 s	TIMEOUT ERROR, Killed. Hard limit reached: 6.000 sec.			
•	large_ra	ange c sequences, X = 30,000	X	TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.	
1.	6.000 s	TIMEOUT ERROR, Killed. Hard	limit	reached: 6.000 sec.	

The PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.