Codility_

Candidate Report: training7JBVA2-V2C

Test Name:

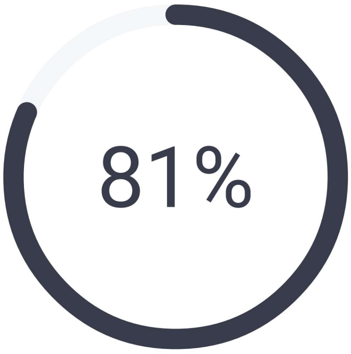
Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
FrogRiverOne Python ⚠	1 min	81%

Total score



Tasks Details

Easy

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

Task Score	Correctness	Performance
81%	100%	60%

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

Solution

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

Task timeline

03:26:35 03:27:18

Code: 03:27:18 UTC, py, final, score: **81**

show code in pop-up

```
# you can write to stdout for debugging purposes,
# print("this is a debug message")

def solution(X, A):
```

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:

```
def solution(X, 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[0] = 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].

```
    return findEarliestTime3(X,A)

def findEarliestTime3(lengthOfBridge:int, leave_fa
    """ Find the earliest time (index of list) to

    # Init a bridge.
    bridge = [False]*(lengthOfBridge+1)
    bridge[0] = True

    # Travel the list
    for i, pos in enumerate(leave_falls):
        bridge[pos] = True
        # Determine a bridge was built or not
        if all(bridge):
            return i
    return -1
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

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

collapse all Example tests

▼ example ✓ OK
example test

1. 0.036 s **OK**

collapse all Correctness tests

▼ simple ✓ OK
simple test

1. 0.036 s **OK**

▼ single ✓ OK
single element

1. 0.040 s **OK**
2. 0.036 s **OK**

▼ extreme_frog ✓ OK
frog never across the river

1. 0.036 s **OK**
2. 0.036 s **OK**
3. 0.036 s **OK**

▼ small_random1 ✓ OK
3 random permutation, X = 50

1. 0.036 s **OK**

▼ small_random2 ✓ OK
5 random permutation, X = 60

1. 0.036 s **OK**

▼ extreme_leaves ✓ OK
all leaves in the same place

1. 0.036 s **OK**
2. 0.036 s **OK**

collapse all

Performance tests

▼ medium_random ✓ OK
6 and 2 random permutations, X = ~5,000

1. 0.080 s **OK**

2. 0.052 s **OK**

▼ medium_range ✓ OK
arithmetic sequences, X = 5,000

1. 0.076 s **OK**

▼ large_random ✓ OK
10 and 100 random permutation, X = ~10,000

1. 0.448 s **OK**

2. 0.152 s **OK**

▼ large_permutation ✗ TIMEOUT ERROR
permutation tests
running time: 0.648 sec., time limit: 0.592 sec.

1. 0.648 s **TIMEOUT ERROR**, running time: 0.648 sec., time limit: 0.592 sec.

2. 6.000 s **TIMEOUT ERROR**, Killed. Hard limit reached: 6.000 sec.

▼ large_range ✗ TIMEOUT ERROR
arithmetic sequences, X = 30,000
running time: 2.936 sec., time limit: 0.336 sec.

1. 2.936 s **TIMEOUT ERROR**, running time: 2.936 sec., time limit: 0.336 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.