

Summary

Timeline

Tasks summary

Task	Time spent	Score
FrogRiverOne Python	119 min	54%

Total score

54%

Tasks Details

Easy	1. FrogRiverOne	Task Score	Correctness	Performance
	Find the earliest time when a frog can jump to the other side of a river.	54%	100%	0%

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.

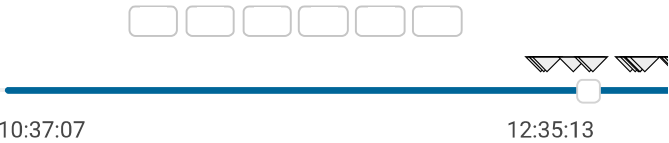
For example, you are given integer X = 5 and array A such that:

```
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:	119 minutes	
Effective time used:	119 minutes	
Notes:	<i>not defined yet</i>	

Task timeline



Code: 12:35:12 UTC, py,

final, score: 54

show code in pop-up

```
1 # you can write to stdout for debugging purposes,
2 # print("this is a debug message")
3
4 # <釐清問題>
```

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].

Copyright 2009–2021 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
5 # TThis is a 跨越河岸的問題
6 # 定義:
7 # - river: 河岸之間的座標(1~X)
8 # - side: 終點與起始點座標分別為X+1和0\
9
10 # - That is, we want to find the earliest momen
11
12 # 目標:
13 # 透過葉子出現的排列組合尋找到目的地(對岸)的最短
14 # (青蛙從座標0開始跳, 欲跳到座標為X+1的對岸(終點
15
16 # X為river的長度(會踩過的葉子數量), N為陣列長度(全部
17 # e.g.
18 # X=5, N=7, A = [1,3,1,4,2,3,5,4]
19 # A[0] = 1 (在第0秒時落葉的座標在1)
20 # A[1] = 3
21 # A[2] = 1
22 # A[3] = 4
23 # A[4] = 2
24 # A[5] = 3
25 # A[6] = 5
26 # A[7] = 4
27
28
29 # <釐清問題2>
30 # 走訪一陣列A, 並用一個長度為X的boolean字典紀錄出現
31 # e.g.
32 # X=5(河面長度, 為程式的終止條件之一), N=7(陣列長度,
33 # A = [1,3,1,4,2,3,5,4](儲存每次葉子掉落的座標位置)
34 # A[0] = 1
35 # A[1] = 3
36 # A[2] = 1
37 # A[3] = 4
38 # A[4] = 2
39 # A[5] = 3
40 # A[6] = 5 (在6秒時, 河面上(座標1~5)皆有葉子
41 # A[7] = 4
42
43 # <定義知識&製作solution><驗證&改進>
44 #
45 # 其它限制:
46 # - N and X are integers within the range [1..10
47 # - each element of array A is an integer within
48 # - 若陣列走訪完葉子橋還沒建好, 則回傳-1
49
50 def solution(X, A):
51     return findEarliestTime(X,A)
52
53 def findEarliestTime(lengthOfBridge:int, leave_fal
54     """ Find the earliest time (index of list) to
55
56     # Init a bridge.
57     bridge = {}
58     for i in range(lengthOfBridge):
59         bridge[i+1] = False
60
61     # Travel the list
62     for i, pos in enumerate(leave_falls):
63         bridge[pos] = True
64
65         if wasBuilt(lengthOfBridge, bridge):
66             return i
67     return -1
68
69 def wasBuilt(lengthOfBridge, bridge):
70     """ Determine a bridge was built or not """
71     for i in range(lengthOfBridge):
72         if bridge[i+1] == False:
73             return False
74     return True
75
76 # def findEarliestTimeFast(lengthOfBridge:int, lea
77
78 # # Init a bridge.
79 # # bridge = {}
80 # # for i in range(lengthOfBridge):
81 # #     bridge[i+1] = False
82
83 # for j in range(lengthOfBridge):
84 #     indexes = []
```

```
85 # # Travel the list
86 #     for i, pos in enumerate(leave_falls):
87 #         if pos == j
88
89 # # Generate test-input
90 # import random
91 # lengthOfBridge = 10 # X
92 # array_length = 10000 # N
93 # leave_falls = []
94 # for i in range(array_length):
95 #     leave_falls.append(random.randint(1,lengthOf
96 # print( "("+ str(lengthOfBridge) + "," + str(leav
97
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

Detected time complexity: **$O(N^2)$**

collapse all		Example tests
▼	example example test	✓ OK
1. 0.036 s OK		
collapse all		Correctness tests
▼	simple simple test	✓ OK
1. 0.036 s OK		
▼	single single element	✓ OK
1. 0.036 s OK		
2. 0.036 s OK		
▼	extreme_frog frog never across the river	✓ OK
1. 0.036 s OK		
2. 0.036 s OK		
3. 0.036 s OK		
▼	small_random1 3 random permutation, X = 50	✓ OK
1. 0.036 s OK		
▼	small_random2 5 random permutation, X = 60	✓ OK
1. 0.036 s OK		
▼	extreme_leaves all leaves in the same place	✓ OK
1. 0.036 s OK		
2. 0.036 s OK		
collapse all		Performance tests
▼		

medium_random 6 and 2 random permutations, X = ~5,000	X TIMEOUT ERROR running time: 0.512 sec., time limit: 0.224 sec.
1. 0.512 s TIMEOUT ERROR , running time: 0.512 sec., time limit: 0.224 sec.	
2. 0.148 s OK	
▼ medium_range arithmetic sequences, X = 5,000	X TIMEOUT ERROR running time: 0.616 sec., time limit: 0.176 sec.
1. 0.616 s TIMEOUT ERROR , running time: 0.616 sec., time limit: 0.176 sec.	
▼ large_random 10 and 100 random permutation, X = ~10,000	X TIMEOUT ERROR running time: 5.752 sec., time limit: 0.592 sec.
1. 5.752 s TIMEOUT ERROR , running time: 5.752 sec., time limit: 0.592 sec.	
2. 0.172 s OK	
▼ large_permutation permutation 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_range arithmetic 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.