Test results - Codility 20/06/16 21:52





Training ticket

Session

ID: training6R8K6G-8AE Time limit: 120 min.

Status: closed

Created on: 2016-06-20 19:33 UTC Started on: 2016-06-20 19:33 UTC Finished on: 2016-06-20 19:44 UTC

Tasks in test

1 | := FrogRiverOne

Correctness

100%

Performance

Task score

100%

Test score 2

100%

100 out of 100 points

1. FrogRiverOne

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

score: 100 of 100

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 a zero-indexed 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:

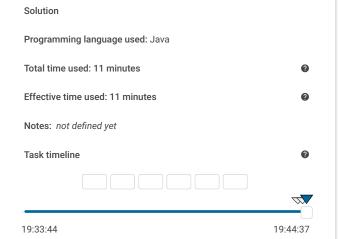
- A[0] = 1
- A[1] = 3
- A[2] = 1
- A[3] = 4A[4] = 2
- A[5] = 3
- A[6] = 5
- A[7] = 4

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 zero-indexed 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.



Code: 19:44:37 UTC, java, final, show code in pop-up score: 100 import java.util.HashMap; import java.util.Map; private static final int NO_RESULT = -1; 6 7 public int solution(int X, int[] A) { 8 Map<Integer, Integer> occurrences = new HashMar for (int i = 0; i < A.length; i++) {</pre> int leaf = A[i]; 10 11 **if** (leaf <= X) { 12 occurrences.putIfAbsent(leaf, i); 13 14 if(occurrences.size() == X) { 15 return occurrences.values().stream(16 17 }

20/06/16 21:52 Test results - Codility

If the frog is never able to jump to the other side of the river, the function

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.

Assume that:

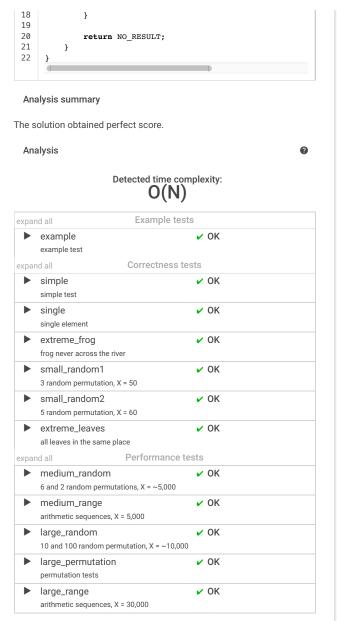
- N and X are integers within the range [1..100,000];
- each element of array A is an integer within the range [1..X].

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(X), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Copyright 2009–2016 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.



Training center