closed

Demo ticket

Session
ID: demo3D7QCG-R7Q
Time limit: 120 min.

Status: closed
Started on: 2014-01-05 06:43 UTC

Score:

100

of 100

1. FrogRiverOne

score: 100 of 100

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

Task description

A small frog wants to get to the other side of a river. The frog is currently located at position 0, and wants to get to position X. Leaves fall from a tree onto the surface of the river.
You are given a non-empty 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 minutes.
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.
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
```

In minute 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.
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. 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.

Solution

Programming language used: C#

Total time used: 4 minutes (?)

Effective time used: 3 minutes (?)

Notes: correct functionality and scalability

Task timeline What is it?

06:43:28 06:46:51

Code: 06:46:51 UTC, cs, final, score: 100.00

```
01. using System;
02. // you can also use other imports, for
    example:
03. // using System.Collections.Generic;
04. class Solution {
05.         public int solution(int X, int[]
    A)
06.         {
07.             // write your code in C# with
    .NET 2.0
08.             var length = A.Length;
09.             if (X < 1 || length < 1 || X >
    100000 || length > 100000)
                throw new
    ArgumentOutOfRangeException();
10.
11.             var expectedSumOfSpots = 0;
12.             for (var count = 1; count <= X;
    count++)
13.                 expectedSumOfSpots +=
    count;
14.
15.             var spotsFound = new bool[X];
16.             var currentSumOfSpots = 0;
17.             for (int count = 0; count <
    length; count++)
18.             {
19.                 if (spotsFound[A[count] -
    1] == false)
20.                     currentSumOfSpots +=
    A[count];
21.
22.                 if (currentSumOfSpots ==
    expectedSumOfSpots)
23.                     return count;
24.
```

```
25.                 spotsFound[A[count] - 1] =
                       true;
26.             }
27.
28.             return -1;
29.         }
30. }
```

Analysis

Detected time complexity:

O(N)

test	time	result
example example test	0.080 s.	**OK**
simple simple test	0.080 s.	**OK**
single single element	0.080 s.	**OK**
extreme_frog frog never across the river	0.080 s.	**OK**
small_random1 3 random permutation, X = 50	0.080 s.	**OK**
small_random2 5 random permutation, X = 60	0.080 s.	**OK**
medium_random 6 and 2 random permutations, X = ~5,000	0.080 s.	**OK**
medium_range arithmetic sequences, X = 5,000	0.080 s.	**OK**
large_random 10 and 100 random permutation, X = ~10,000	0.100 s.	**OK**
large_permutation permutation tests	0.100 s.	**OK**
large_range arithmetic sequences, X = 30,000	0.090 s.	**OK**
extreme_leaves all leaves in the same place	0.080 s.	**OK**