

AVAILABLE  
LESSONS:*Lesson 1*  
Iterations*Lesson 2*  
Arrays*Lesson 3*  
Time Complexity*Lesson 4*  
**Counting  
Elements***Lesson 5*  
Prefix Sums*Lesson 6*  
Sorting*Lesson 7*  
Stacks and  
Queues*Lesson 8*  
Leader*Lesson 9*  
Maximum slice  
problem*Lesson 10*  
Prime and  
composite  
numbers*Lesson 11*

PAINLESS

## FrogRiverOne

START

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

Programming language: 

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] = 4
A[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:

```
int solution(int X, vector<int> &A);
```

that, given a non-empty zero-indexed array  $A$  consisting of  $N$

Sieve of  
Eratosthenes

*Lesson 12*  
Euclidean  
algorithm

*Lesson 13*  
Fibonacci  
numbers

*Lesson 14*  
Binary search  
algorithm

*Lesson 15*  
Caterpillar  
method

*Lesson 16*  
Greedy  
algorithms

*Lesson 17*  
Dynamic  
programming

*Lesson 90*  
Tasks from  
Indeed Prime  
2016 challenge

*Lesson 99*  
Future training

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.

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

## For programmers

[Lessons](#) [Challenges](#)  
[Terms](#) [FAQ](#)

## For companies

[Tour](#) [About us](#)  
[Pricing](#) [Jobs](#)  
[Blog](#) [Terms](#)  
[Privacy](#) [Cookies](#)  
[API](#)

## Sign up for our newsletter:

Information about upcoming challenges, solutions and lessons directly in your inbox.

[Sign up](#)

## Social:

[f](#) [t](#) [in](#)

## Contact us:

For customer support queries:

UK +44 (0) 208 970 78 68

US 1-415-466-8085  
[support@codility.com](mailto:support@codility.com)

For sales queries:

UK +44 (0) 208 970 78 67

US 1-415-466-8085  
[sales@codility.com](mailto:sales@codility.com)

---

© 2009–2016 Codility Ltd., registered in England and Wales (No. 7048726). VAT ID GB981191408. Registered office: 107 Cheapside, London EC2V 6DN