



CodeCheck Report: trainingDFK5K7-2HZ

[Check out Codility training tasks](#)

Test Name:

- Summary
- Timeline
- AI Assistant Transcript

Tasks summary

Task	Time spent	Score
FrogRiverOne	15 min	100%
C++		

Total score



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.	100%	100%	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 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.

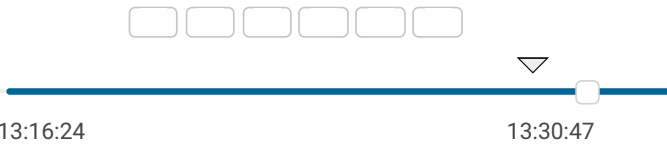
For example, you are given integer X = 5 and array A such that:

```
A[0] = 1
A[1] = 3
A[2] = 1
A[3] = 4
A[4] = 2
A[5] = 3
```

Solution

Programming language used:	C++
Total time used:	15 minutes
Effective time used:	15 minutes
Notes:	not defined yet

Task timeline



Code: 13:30:46 UTC, cpp, [show code in pop-up](#)
final, score: 100

```
1 // you can use includes, for example:
2 // #include <algorithm>
3 #include <unordered_set>
```

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 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].

Test results - Codility

```
4
5 // you can write to stdout for debugging purposes
6 // cout << "this is a debug message" << endl;
7
8 int solution(int X, vector<int> &A) {
9     //0 -> x+1
10    //k시간에 a[k] 위치에 잎이 떨어짐
11    //떨어진 잎은 그 자리에 있음
12    //1칸씩밖에 못 점프함
13    //몇초에 x에 도달하는가?
14    //결국 못 건너면 -1 반환
15
16    int answer = -1;
17
18    unordered_set<int> s;
19    for (int i = 0; i < A.size(); i++) {
20        s.insert(A[i]);
21        if (s.size() == X) {
22            return i;
23        }
24    }
25
26    return answer;
27
28 }
```

Analysis summary

The solution obtained perfect score.

Analysis

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

Example tests	
▶ example	✓ OK
example test	
Correctness tests	
▶ simple	✓ OK
simple test	
▶ single	✓ OK
single element	
▶ extreme_frog	✓ OK
frog never across the river	
▶ small_random1	✓ OK
3 random permutation, X = 50	
▶ small_random2	✓ OK
5 random permutation, X = 60	
▶ extreme_leaves	✓ OK
all leaves in the same place	
Performance tests	
▶ medium_random	✓ OK
6 and 2 random permutations, X = ~5,000	
▶ medium_range	✓ OK
arithmetic sequences, X = 5,000	
▶ large_random	✓ OK
10 and 100 random permutation, X = ~10,000	
▶	

large_permutation	✓ OK
permutation tests	
▶ large_range	✓ OK
arithmetic sequences, X = 30,000	