Congratulations

You have completed a Codility training test.

Tweet this!
I scored 100% in #cpp on @Codility!
https://codility.com/demo/take-sample-test/frog_river_one/

Sign up for our newsletter!

Like us on Facebook!

g center

ility training tasks

Training ticket

Session

ID: trainingYG9RAS-VMX
Time limit: 120 min.

Status: closed

Created on: 2016-07-24 17:08 UTC
Started on: 2016-07-24 17:08 UTC
Finished on: 2016-07-24 17:21 UTC

Tasks in test	Correctness	Performance	Task score
1 FrogRiverOne Submitted in: C++	100%	100%	100%

100%

100 out of 100 points

EASY

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] = 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

Solution

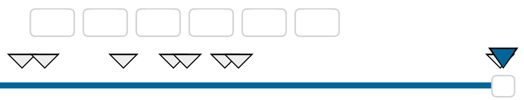
Programming language used: C++

Total time used: 13 minutes

Effective time used: 13 minutes

Notes: *not defined yet*

Task timeline



17:08:33 17:21:13

Code: 17:21:12 UTC, cpp, final, score: **100**

show code in pop-up

```
#include <queue>
int solution(int X, vector<int> &A) {
    priority_queue<int, vector<int>, greater<int>> pq;
    const int D = 1;
    const int N = A.size();
    int cur = 0;
    if (cur + D > X) return 0;
```

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 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.

```

    for (int i = 0; i < N; ++i) {
        if (A[i] <= cur) continue;

        pq.push(A[i]);
        while (!pq.empty() && pq.top() <= cur + D) {
            cur = pq.top(); pq.pop();
        }

        if (cur + D > X) return i;
    }
    return -1;
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity:
O(N)

expand all	Example tests	
▶	example example test	✔ OK

expand all	Correctness tests	
▶	simple simple test	✔ OK
▶	single single element	✔ OK
▶	extreme_frog frog never across the river	✔ OK
▶	small_random1 3 random permutation, X = 50	✔ OK
▶	small_random2 5 random permutation, X = 60	✔ OK
▶	extreme_leaves all leaves in the same place	✔ OK

expand all	Performance tests	
▶	medium_random 6 and 2 random permutations, X = ~5,000	✔ OK
▶	medium_range arithmetic sequences, X = 5,000	✔ OK
▶	large_random 10 and 100 random permutation, X = ~10,000	✔ OK
▶	large_permutation permutation tests	✔ OK
▶	large_range arithmetic sequences, X = 30,000	✔ OK

Training center