codility

Training center

y training tasks

Congratulations

You have completed a Codility training test.

Sign up for our newsletter!

Like us on Facebook!

Training ticket

Session

ID: trainingVTM3ZE-TZT
Time limit: 120 min.

Status: closed

Created on: 2016-06-05 09:01 UTC
Started on: 2016-06-05 09:01 UTC
Finished on: 2016-06-05 09:02 UTC

Tasks in test	Correctness	Performance	Task score
1 | MaxCounters Submitted in: Java	100%	60%	77%

77%

77 out of 100 points

MEDIUM

1. MaxCounters

score: 77 of 100

Calculate the values of counters after applying all alternating operations: increase counter by 1; set value of all counters to current maximum.

Task description

You are given N counters, initially set to 0, and you have two possible operations on them:

- *increase(X)* − counter X is increased by 1,
- *max counter* − all counters are set to the maximum value of any counter.

A non-empty zero-indexed array A of M integers is given. This array represents consecutive operations:

- if A[K] = X, such that 1 ≤ X ≤ N, then operation K is increase(X),
- if A[K] = N + 1 then operation K is max counter.

For example, given integer N = 5 and array A such that:

```
A[0] = 3
A[1] = 4
A[2] = 4
A[3] = 6
A[4] = 1
A[5] = 4
A[6] = 4
```

the values of the counters after each consecutive operation will be:

```
(0, 0, 1, 0, 0)
(0, 0, 1, 1, 0)
(0, 0, 1, 2, 0)
(2, 2, 2, 2, 2)
(3, 2, 2, 2, 2)
(3, 2, 2, 3, 2)
(3, 2, 2, 4, 2)
```

The goal is to calculate the value of every counter after all operations.

Solution

Programming language used: Java

Total time used: 2 minutes

Effective time used: 2 minutes

Notes: *not defined yet*

Task timeline

09:01:01 09:02:03

Code: 09:02:03 UTC, java, final, score: 77

show code in pop-up

```
1  // you can also use imports, for example:
2  // import java.util.*;
3
4  // you can write to stdout for debugging purposes, e.g.
5  // System.out.println("this is a debug message");
6
7  class Solution {
8      public int[] solution(int N, int[] A) {
9          int[] a = new int[N];
10         int c = 0;
11         int max = 0;
12         for (int i=0;i<A.length;i++) {
13           c = A[i];
```

Write a function:

```
class Solution { public int[] solution(int N, int[] A); }
```

that, given an integer N and a non-empty zero-indexed array A consisting of M integers, returns a sequence of integers representing the values of the counters.

The sequence should be returned as:

- a structure Results (in C), or
- a vector of integers (in C++), or
- a record Results (in Pascal), or
- an array of integers (in any other programming language).

For example, given:

```
A[0] = 3
A[1] = 4
A[2] = 4
A[3] = 6
A[4] = 1
A[5] = 4
A[6] = 4
```

the function should return [3, 2, 2, 4, 2], as explained above.

Assume that:

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

Complexity:

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

Elements of input arrays can be modified.


```
14            if (c > 0 && c <=N) {
15              a[c-1]++;
16              if (a[c-1] > max) {
17                max = a[c-1];
18              }
19            } else if (c == (N+1)) {
20              for (int k=0;k<a.length;k++) {
21                a[k] = max;
22              }
23            }
24          }
25          return a;
26      }
27  }
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

Detected time complexity:
O(N*M)

expand all	Example tests	
▶	example example test	✔ OK
expand all	Correctness tests	
▶	extreme_small all max_counter operations	✔ OK
▶	single only one counter	✔ OK
▶	small_random1 small random test, 6 max_counter operations	✔ OK
▶	small_random2 small random test, 10 max_counter operations	✔ OK
expand all	Performance tests	
▶	medium_random1 medium random test, 50 max_counter operations	✔ OK
▶	medium_random2 medium random test, 500 max_counter operations	✔ OK
▶	large_random1 large random test, 2120 max_counter operations	✔ OK
▶	large_random2 large random test, 10000 max_counter operations	✘ TIMEOUT ERROR running time: 5.36 sec., time limit: 5.19 sec.
▶	extreme_large all max_counter operations	✘ TIMEOUT ERROR running time: >14.00 sec., time limit: 8.15 sec.