Codility_

CodeCheck Report: trainingM4G67B-9YH

Test Name:

Check out Codility training tasks

Summary   Timeline

Tasks summary

Task	Time spent	Score
MaxCounters ⚠ C#	2 min	100%

Total score

100%

Tasks Details

Medium

1. MaxCounters

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

Task Score	Correctness	Performance
100%	100%	100%

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 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:

Solution

Programming language used:	C#
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	*not defined yet*

Task timeline

```
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.

Write a function:

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

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

Result array should be returned as an array of integers.

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.

Write an **efficient** algorithm for the following assumptions:

- 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].

04:05:13 04:06:37

Code: 04:06:36 UTC, cs, final, score: **100**

show code in pop-up

```
using System;

/* Lesson 4.2 - Max Counter
 * Paulo Santos
 * 24.Nov.2022
 */
class Solution {
    public int[] solution(int N, int[] A) {

                /*
                 * Check the inputs
                 */
        if (A == null)
            throw new ArgumentNullException("A is
                if (N < 0)
                        throw new ArgumentOutOfRan

        var cnt = new int[N]; // counters
        var mct = new int[N]; // indicate the need
        var max = 0;          // current max count
        var mcc = 0;          // number of times t
        var mud = 0;          // max counter for u
        for(var i = 0; i < A.Length; i++) {
            /*
             * Check the command to see if it's to
             */
            var ord = A[i] - 1;
            if (0 <= ord && ord <= (N - 1)) {
                /*
                 * Check the need to max the count
                 */
                if (mct[ord] != mcc) {
                    mct[ord] = mcc;
                    cnt[ord] = mud;
                }
                max = Math.Max(max, ++cnt[ord]);
            }
            else
            {
                /*
                 * Add the count to max the counte
                 */
                mcc++;
                mud = max;
            }
        }

        /*
         * Adjust the counters that were not maxed
         */
        for (var i = 0 ; i < cnt.Length; i++)
            if (mct[i] != mcc)
                cnt[i] = mud;

        return cnt;
    }
}
```

Analysis summary

The solution obtained perfect score.

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	✓ OK
▶	extreme_large all max_counter operations	✓ OK