Codility_

Candidate Report: training9FJ659-C73

Check out Codility training tasks

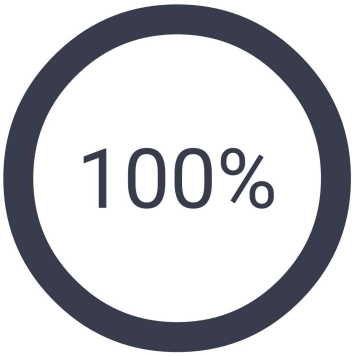
Test Name:

Summary Timeline Feedback

Tasks summary

Task	Time spent	Score
MaxCounters C#	1 min	100%

Total score



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:

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

Solution

Programming language used:	C#
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline

19:41:25 19:41:44

Code: 19:41:44 UTC, cs, final, score: **100**

show code in pop-up

```
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].


```
1  using System;
2  // you can also use other imports, for example:
3  // using System.Collections.Generic;
4
5  // you can write to stdout for debugging purposes, e.g.
6  // Console.WriteLine("this is a debug message");
7
8  class Solution {
9     public int[] solution(int N, int[] A)
10        {
11
12            int[] counter = new int[N];
13            int baseMinimum = 0;
14            int possibleMinimum = 0;
15            int index = 0;
16
17            for (int i = 0; i < A.Length; i++)
18            {
19                index = A[i] - 1;
20
21                if (index == N)
22                {
23                    baseMinimum = possibleMinimum;
24                }
25                else
26                {
27                    counter[index] = Math.Max(counter[index
28                    counter[index] += 1;
29
30                    if (possibleMinimum < counter[index])
31                    {
32                        possibleMinimum = counter[index];
33                    }
34                }
35            }
36
37            for (int i = 0; i < counter.Length; i++)
38            {
39                counter[i] = Math.Max(counter[i], baseMinim
40            }
41
42            return counter;
43        }
44 }
```

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	✓ OK

	operations	
▶	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