codility

Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

100 out of 100 points

100%

Tasks in Test

	Time Spent	Task Score
MaxCounters Submitted in: Python	10 min	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:

```
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)
```

Solution

Programming language used:	Python
Total time used:	10 minutes
Effective time used:	10 minutes
Notes:	*not defined yet*

Task timeline

12:33:01 12:42:32

Code: 12:42:31 UTC, py, final, score: **100** show code in pop-up

```
1  def solution(n, a):
2      counters = [0 for _ in range(n)]
3      lower = 0
4      higher = 0
5      for command in a:
6          i = command - 1
```

```
(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:

```
def solution(N, 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].


```
        if command < n + 1:
            value = max(counters[i], lower) + 1
            higher = max(higher, value)
            counters[i] = value
        else:
            lower = higher
    for i in range(n):
        if counters[i] < lower:
            counters[i] = lower
    return counters
```

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