Codility_

CodeCheck Report: trainingVFG8TB-JEU

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxCounters Python ⚠	20 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:

```
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:

```
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.

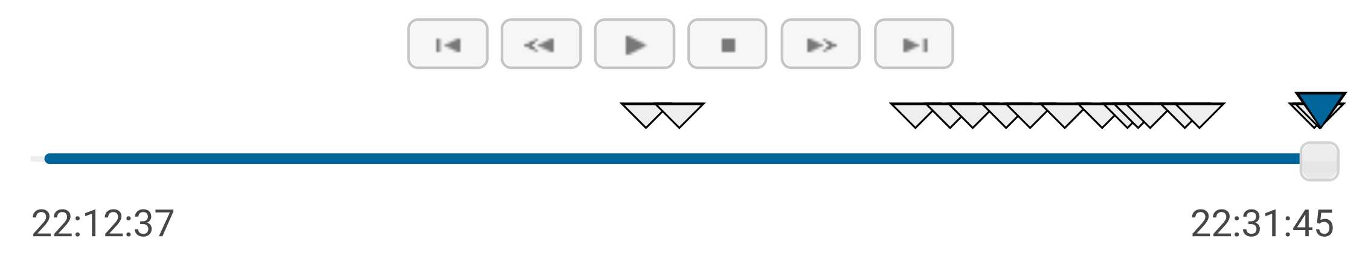
Write an **efficient** algorithm for the following assumptions:

- 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].

Solution

Programming language used:	Python	
Total time used:	20 minutes	?
Effective time used:	20 minutes	?
Notes:	*not defined yet*	

Task timeline



Code: 22:31:44 UTC, py, final, score: 100

show code in pop-up

```python
# you can write to stdout for debugging purposes, e.g.
# print("this is a debug message")

def solution(N, A):
    # write your code in Python 3.6

    if max(A) > N+1:
        return -1
    else:
        total_resets = 0
        B = {}
        max_ = 0
        for el in A:
            if el < N+1:
                if el not in B.keys():
                    B[el] = 1
                else:
                    B[el] += 1
                max_ = max(max_, B[el])
            else:
                total_resets += max_
                max_ = 0
                B = {}

        C = [0] * N
        for i in range(len(A)-1, -1, -1):
            if A[i] != N+1:
                C[A[i]-1] += 1
            else:
                break

        D = [C[i] + total_resets for i in range(N)]

        return D
```

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