Codility_

## Candidate Report: trainingRHPQM2-X8U

Check out Codility training tasks

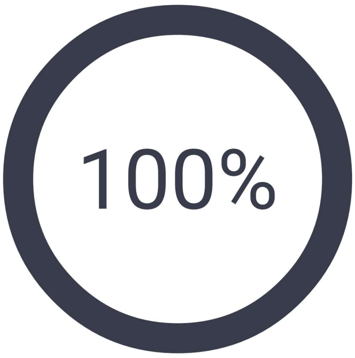
Test Name:

Summary Review (0) Timeline

### Tasks summary

| Task | Time spent | Score |
|---|---|---|
| MaxCounters<br>Java 8 | 1 min | 100% |

### Total score



## Tasks Details

### 1. MaxCounters

Medium

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:

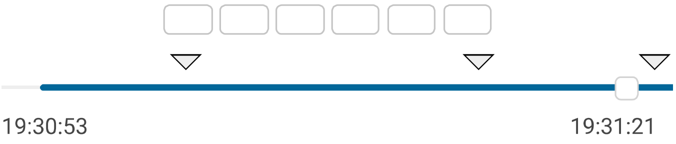
```
    A[0] = 3
    A[1] = 4
    A[2] = 4
    A[3] = 6
    A[4] = 1
    A[5] = 4
    A[6] = 4
```

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

### Solution

| | |
|---|---|
| Programming language used: | Java 8 |
| Total time used: | 1 minutes |
| Effective time used: | 1 minutes |
| Notes: | *not defined yet* |

### Task timeline



Code: 19:31:20 UTC, java, final, score: **100**

show code in pop-up

```
// you can also use imports, for example:
// import java.util.*;

// you can write to stdout for debugging purposes, e.g.
```

```
    (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].

```
// System.out.println("this is a debug message");

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

    final int condition = N + 1;
    int currentMax = 0;
    int lastUpdate = 0;
    int countersArray[] = new int[N];

    for (int iii = 0; iii < A.length; iii++) {
      int currentValue = A[iii];
      if (currentValue == condition) {
        lastUpdate = currentMax;
      } else {
        int position = currentValue - 1;
        if (countersArray[position] < lastUpdate)
          countersArray[position] = lastUpdate
        else
          countersArray[position]++;

        if (countersArray[position] > currentMax)
          currentMax = countersArray[position];
        }
      }

    }

    for (int iii = 0; iii < N; iii++) {
       if (countersArray[iii] < lastUpdate)
         countersArray[iii] = lastUpdate;
    }

    return countersArray;
  }
}
```

## Analysis summary

The solution obtained perfect score.

## Analysis ?

Detected time complexity: **O(N + M)**

| expand all | Example tests | |
|---|---|---|
| ▶ | example<br>example test | ✓ OK |
| expand all | Correctness tests | |
| ▶ | extreme_small<br>all max_counter operations | ✓ OK |
| ▶ | single<br>only one counter | ✓ OK |
| ▶ | small_random1<br>small random test, 6 max_counter operations | ✓ OK |
| ▶ | small_random2<br>small random test, 10 max_counter operations | ✓ OK |
| expand all | Performance tests | |
| ▶ | medium_random1<br>medium random test, 50 max_counter operations | ✓ OK |
| ▶ | medium_random2<br>medium random test, 500 max_counter | ✓ OK |

| | | |
|---|---|---|
| | operations | |
| ▶ | large_random1<br>large random test, 2120 max_counter operations | ✓ OK |
| ▶ | large_random2<br>large random test, 10000 max_counter operations | ✓ OK |
| ▶ | extreme_large<br>all max_counter operations | ✓ OK |

The PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.