Codility_

CodeCheck Report: trainingGR5M27-XW2

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
MaxCounters C++	5 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.

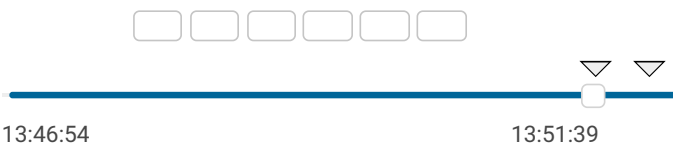
For example, given integer N = 5 and array A such that:

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

Solution

Programming language used:	C++
Total time used:	5 minutes
Effective time used:	5 minutes
Notes:	*not defined yet*

Task timeline



13:46:54 13:51:39

Code: 13:51:39 UTC, cpp, 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:

```
vector<int> solution(int N, vector<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 a vector 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].


```cpp
// you can use includes, for example:
// #include <algorithm>

// you can write to stdout for debugging purpo
// cout << "this is a debug message" << endl;

void addAll(vector<int>& v) {
  int max = 0;
  for (auto& e : v) {
    if (max < e) {
      max = e;
    }
  }

  for (auto& e : v) {
    e = max;
  }

  return;
}

vector<int> solution(int N, vector<int>& A) {
  int maxCounter = N + 1;
  vector<int> v(N, 0);

  int currentMax = 0;
  int lastUpdate = 0;

  for (int i = 0; i < A.size(); i++) {
    int target = A[i];

    if (target == maxCounter) {
      lastUpdate = currentMax;
    } else {
        if(v[target - 1] < lastUpdate){
            v[target - 1] = lastUpdate + 1;
        } else {
            v[target - 1]++;
        }

        currentMax = max(currentMax, v[target-
    }
  }

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

  return v;
}
```

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