Codility_

CodeCheck Report: trainingTF33RR-2PB

Test Name:

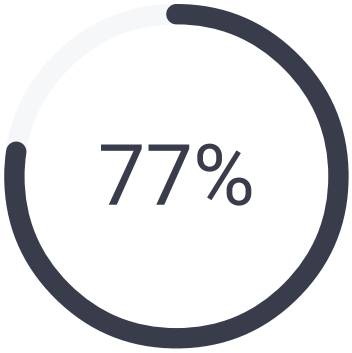
Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
MaxCounters C++	10 min	77%

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
77%	100%	60%

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.

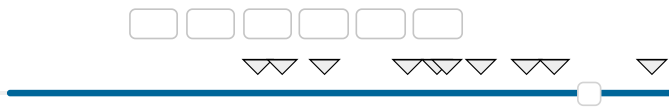
For example, given integer N = 5 and array A such that:

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

Solution

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

Task timeline



13:32:31 13:42:20

Code: 13:42:20 UTC, cpp, final, score: 77 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);

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

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

    return v;
}
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

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	✘ TIMEOUT ERROR running time: 0.372 sec., time limit: 0.100 sec.
1.	0.372 s	**TIMEOUT ERROR**, running time: 0.372 sec., time limit: 0.100 sec.
▼	extreme_large all max_counter operations	✘ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
1.	6.000 s	**TIMEOUT ERROR**, Killed. Hard limit reached: 6.000 sec.
2.	0.008 s	**OK**