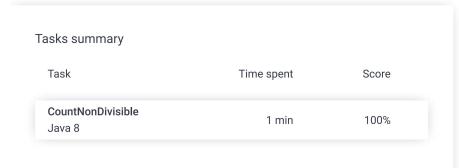
Codility_

Candidate Report: trainingFGU4UV-UY4

Check out Codility training tasks

Test Name:

Summary Review (0) Timeline





Tasks Details

1. CountNonDivisible

Calculate the number of elements of an array that are not divisors of each element.

Task Score

Correctness

Performance

100%

100%

Task description

You are given an array A consisting of N integers.

For each number A[i] such that $0 \le i < N$, we want to count the number of elements of the array that are not the divisors of A[i]. We say that these elements are non-divisors.

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

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

For the following elements:

- A[0] = 3, the non-divisors are: 2, 6,
- A[1] = 1, the non-divisors are: 3, 2, 3, 6,
- A[2] = 2, the non-divisors are: 3, 3, 6,
- A[3] = 3, the non-divisors are: 2, 6,
- A[4] = 6, there aren't any non-divisors.

Write a function:

class Solution { public int[] solution(int[] A); }

that, given an array A consisting of N integers, returns a sequence of integers representing the amount of non-divisors.

Result array should be returned as an array of integers.

Solution

100%

Test results - Codility

Programming language used:

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: not defined yet

Task timeline

09:49:47

score: 100



Code: 09:50:30 UTC, java, final, 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.
- // System.out.println("this is a debug message"); 5 import java.util.HashMap;
- class Solution {

09:50:30

For example, given:

A[0] = 3 A[1] = 1A[2] = 2

A[3] = 3

A[4] = 6

the function should return [2, 4, 3, 2, 0], as explained above.

Write an efficient algorithm for the following assumptions:

- N is an integer within the range [1..50,000];
- each element of array A is an integer within the range [1..2 * N].

Copyright 2009–2020 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
public int[] solution(int[] A) {
9
10
                 int N = A.length;
11
                 HashMap<Integer, Integer> count = new HashMap
12
13
                 for (int i : A) {
                     Integer key = count.get(i);
14
                      if (key != null) {
15
16
                          count.put(i, key + 1);
17
                     } else {
                          count.put(i, 1);
18
19
                     }
20
                 }
21
22
                 HashMap<Integer, Integer> divs = new HashMap<>
                 for (Integer n : count.keySet()) {
23
24
                     int sum = 0;
25
                     int j = 1;
                     while (j * j <= n) {
26
                         if (n % j == 0) {
27
                              if (count.containsKey(j)) {
28
29
                                  sum += count.get(j);
30
                              //find n = j*k cases to add both t
31
                              int k = n / j;
32
33
                              if (k != j) {
34
                                  if (count.containsKey(k)) {
35
                                      sum += count.get(k);
36
37
                              }
38
                          }
39
                          j++;
                     }
40
41
42
                      divs.put(n, N - sum);
                 }
43
44
45
                 for (int i = 0; i < A.length; i++) {
                     A[i] = divs.get(A[i]);
46
47
48
                 return A;
49
50
             }
51
     }
```

Analysis summary

The solution obtained perfect score.

Analysis 👩

Detected time complexity: O(N * log(N))

expand all	Example tests	
example example test	√ OK	
expand all	Correctness tests	
extreme_simple extreme simple	√ OK	
double two elements	√ OK	
simple simple tests	√ OK	
primes prime numbers	√ OK	
small_random	✓ OK pers, length = 100	
expand all	Performance tests	

Test results - Codility

•	medium_random medium, random numbers length = 5,000	√ OK
•	large_range 1, 2,, N, length = ~20,000	√ OK
•	large_random large, random numbers, length = ~30,000	√ OK
•	large_extreme large, all the same values, length = 50,000	√ 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.