Codility_

## Candidate Report: trainingVG7FPY-DYJ

Check out Codility training tasks

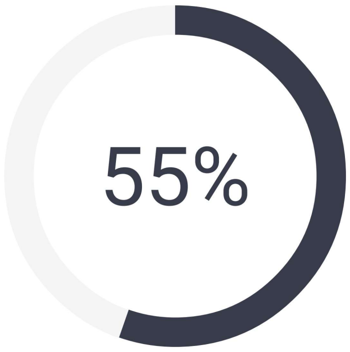
Test Name:

Summary Review (0) Timeline

### Tasks summary

| Task | Time spent | Score |
|---|---|---|
| CountNonDivisible<br>Java 8 | 15 min | 55% |

### Total score



## Tasks Details

Medium

### 1. CountNonDivisible

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

| Task Score | Correctness | Performance |
|---|---|---|
| 55% | 100% | 0% |

### Task description

You are given an array A consisting of N integers.

For each number A[i] such that 0 ≤ 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

| | |
|---|---|
| Programming language used: | Java 8 |
| Total time used: | 15 minutes |
| Effective time used: | 15 minutes |
| Notes: | *not defined yet* |

### Task timeline

09:24:02 09:38:46

Code: 09:38:46 UTC, java, final, score: **55**

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");

class Solution {
```

For example, given:

```
    A[0] = 3
    A[1] = 1
    A[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].

```java
    public int[] solution(int[] A) {
        // write your code in Java SE 8
        int[] result = new int[A.length];
        for(int i=0; i<A.length; i++){
            int count = 0;
            for(int k=0; k<A.length; k++){
                if(i!=k)
                {
                    if(A[i]%A[k]!=0)
                        count++;
                }
            }
            result[i] = count;
        }
        return result;
    }
}
```

## Analysis summary

The following issues have been detected: timeout errors.

## Analysis

Detected time complexity: **O(N ** 2)**

| expand all | Example tests | |
|---|---|---|
| ▶ | example<br>example test | ✓ OK |
| expand all | **Correctness tests** | |
| ▶ | extreme_simple<br>extreme simple | ✓ OK |
| ▶ | double<br>two elements | ✓ OK |
| ▶ | simple<br>simple tests | ✓ OK |
| ▶ | primes<br>prime numbers | ✓ OK |
| ▶ | small_random<br>small, random numbers, length = 100 | ✓ OK |
| expand all | **Performance tests** | |
| ▶ | medium_random<br>medium, random numbers length = 5,000 | ✗ TIMEOUT ERROR<br>running time: 1.224 sec., time limit: 0.304 sec. |
| ▶ | large_range<br>1, 2, ..., N, length = ~20,000 | ✗ TIMEOUT ERROR<br>Killed. Hard limit reached: 7.000 sec. |
| ▶ | large_random<br>large, random numbers, length = ~30,000 | ✗ TIMEOUT ERROR<br>Killed. Hard limit reached: 7.000 sec. |
| ▶ | large_extreme<br>large, all the same values, length = 50,000 | ✗ TIMEOUT ERROR<br>Killed. Hard limit reached: 8.000 sec. |