Codility_

CodeCheck Report: trainingBH6Y7P-AY7

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
CountNonDivisible Python ⚠	6 min	100%

Total score

100%

Tasks Details

1. **CountNonDivisible**

Medium

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

Task Score	Correctness	Performance
100%	100%	100%

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
```

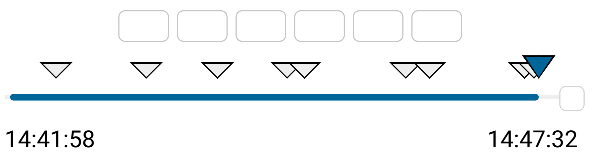
For the following elements:

- 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,

Solution

Programming language used:	Python	
Total time used:	6 minutes	?
Effective time used:	6 minutes	?
Notes:	*not defined yet*	

Task timeline ?



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

Write a function:

```
def solution(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.

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].

Code: 14:47:32 UTC, py, final, score: **100**

show code in pop-up

```
# Source:
# https://github.com/Dineshkarthik/codilit

# you can write to stdout for debugging pu
# print("this is a debug message")

def solution(A):
    # Implement your solution here
    # pass

    _dict = {}
    maxA = 0

    for a in A:
        _dict[a] = _dict.get(a, 0) + 1  #
        maxA = max(a, maxA)
    ND = [len(A) - 1] * (maxA + 1)

    for b in _dict.keys():
        ND[b] -= (_dict[b] - 1)
        m = b * 2
        while m <= maxA:
            ND[m] -= _dict[b]
            m += b
    result = []

    for a in A:
        result += [ND[a]]

    return result
```

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 small, random numbers, length = 100	✓ OK

expand all	Performance tests	
▶	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