Codility_

CodeCheck Report: trainingNQGPUM-UVQ

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
CommonPrimeDivisors ⚠ C#	2 min	100%

Total score

100%

Tasks Details

Medium

1.
CommonPrimeDivisors

Check whether two numbers have the same prime divisors.

Task Score	Correctness	Performance
100%	100%	100%

Task description

A *prime* is a positive integer X that has exactly two distinct divisors: 1 and X. The first few prime integers are 2, 3, 5, 7, 11 and 13.

A prime D is called a *prime divisor* of a positive integer P if there exists a positive integer K such that D * K = P. For example, 2 and 5 are prime divisors of 20.

You are given two positive integers N and M. The goal is to check whether the sets of prime divisors of integers N and M are exactly the same.

For example, given:

- N = 15 and M = 75, the prime divisors are the same: {3, 5};
- N = 10 and M = 30, the prime divisors aren't the same: {2, 5} is not equal to {2, 3, 5};

Solution

Programming language used:	C#	
Total time used:	2 minutes	?
Effective time used:	2 minutes	?
Notes:	*not defined yet*	

Task timeline ?

13:24:14 13:25:26

- N = 9 and M = 5, the prime divisors aren't the same: {3} is not equal to {5}.

Write a function:

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

that, given two non-empty arrays A and B of Z integers, returns the number of positions K for which the prime divisors of A[K] and B[K] are exactly the same.

For example, given:

```
A[0] = 15   B[0] = 75
A[1] = 10   B[1] = 30
A[2] = 3    B[2] = 5
```

the function should return 1, because only one pair (15, 75) has the same set of prime divisors.

Write an **efficient** algorithm for the following assumptions:

- Z is an integer within the range [1..6,000];
- each element of arrays A and B is an integer within the range [1..2,147,483,647].

Code: 13:25:25 UTC, cs, final, score: **100**

show code in pop-up

```
using System;

/**
 * 12.2 - Common Prime Divisors
 * Paulo Santos
 * 27.Dec.2022
 */
class Solution {
    public int solution(int[] A, int[] B) {

        var res = 0;
        for(var i = 0; i < A.Length; i++) {
            res += CheckDiv(A[i], B[i]);
        }
        return res;

    }

    private int CheckDiv(int a, int b) {
        if (a == b) return 1;

        var div = GDC(a, b);
        if (a == 1 || b == 1 || div == 1) return 0

        while (a != 1) {
            var tstA = GDC(a, div);
            if (tstA == 1) break;
            a /= tstA;
        }
        if (a != 1) return 0;

        while (b != 1) {
            var tstB = GDC(b, div);
            if (tstB == 1) break;
            b /= tstB;
        }
        if (b != 1) return 0;

        return 1;
    }

    private int GDC(int a, int b) {
        if (a % b == 0) return b;
        return GDC(b, a % b);
    }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(Z * log(max(A)**

```
+
max(B))**2)
```

expand all	Example tests
▶ example example test	✓ OK
expand all	**Correctness tests**
▶ extreme extreme test with small values	✓ OK
▶ simple_1 simple test with small values	✓ OK
▶ simple_2 simple test with small values	✓ OK
▶ primes powers of primes	✓ OK
▶ small_primes small primes	✓ OK
▶ small_all_pairs all pairs 1-10, length = 100	✓ OK
▶ small_random small random test, length = 100	✓ OK
expand all	**Performance tests**
▶ large_all_pairs all pairs 1-70, length = ~5,000	✓ OK
▶ large_random large random tests, length = ~6,000	✓ OK
▶ many_factors factorial test	✓ OK
▶ many_factors2 factorial test	✓ OK
▶ big_powers powers of 2 and 3	✓ OK
▶ extreme_maximal extreme test with maximal values	✓ OK