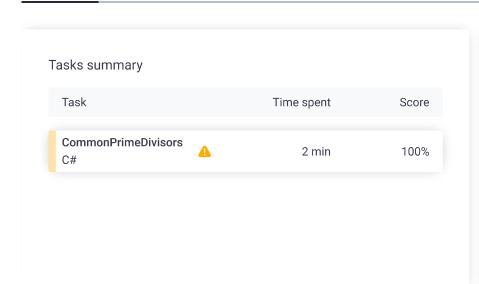
Codility_

CodeCheck Report: trainingNQGPUM-UVQ

Test Name:

Summary Timeline





Check out Codility training tasks

Tasks Details

1.

Medium

CommonPrimeDivisors

Check whether two numbers have the same prime divisors.

Task Score

Correctness 100%

Performance

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].

Copyright 2009–2022 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 13:25:25 UTC, cs, show code in pop-up final, score: 100

```
1
     using System;
 2
3
      * 12.2 - Common Prime Divisors
 4
 5
      * Paulo Santos
      * 27.Dec.2022
 6
 7
      */
8
     class Solution {
         public int solution(int[] A, int[] B) {
9
10
11
             var res = 0;
12
             for(var i = 0; i < A.Length; i++) {</pre>
13
                 res += CheckDiv(A[i], B[i]);
14
15
             return res;
16
17
         }
18
         private int CheckDiv(int a, int b) {
19
20
             if (a == b) return 1;
21
22
             var div = GDC(a, b);
             if (a == 1 || b == 1 || div == 1) return 0
23
24
25
             while (a != 1) {
                 var tstA = GDC(a, div);
26
27
                 if (tstA == 1) break;
28
                 a /= tstA;
29
             }
30
             if (a != 1) return 0;
31
             while (b != 1) {
32
33
                 var tstB = GDC(b, div);
                 if (tstB == 1) break;
34
35
                 b /= tstB;
36
             if (b != 1) return 0;
37
38
39
             return 1;
40
         }
41
42
         private int GDC(int a, int b) {
             if (a % b == 0) return b;
43
44
             return GDC(b, a % b);
45
         }
46
     }
```

Analysis summary

The solution obtained perfect score.

Analysis

O(Z * log(max(A)

Detected time complexity:

+ max(B))**2)

 ▶ example example test expand all Correctness tests ▶ extreme	
expand all Correctness tests extreme extreme extreme test with small values simple_1 simple test with small values simple test with small values simple_2 simple test with small values primes powers of primes	
 ▶ extreme extreme test with small values ▶ simple_1	
extreme test with small values Simple_1	
 ▶ simple_1	
simple test with small values ▶ simple_2	
 ▶ simple_2	
simple test with small values ▶ primes powers of primes ✓ OK	
▶ primes	
powers of primes	
► small_primes ✓ OK	
small primes	
► small_all_pairs ✓ OK	
all pairs 1-10, length = 100	
► small_random ✓ OK	
small random test, length = 100	
expand all Performance tests	
▶ large_all_pairs ✓ OK	
all pairs 1-70, length = ~5,000	
large_random ✓ OK large random tests, length = ~6,000	
► many_factors ✓ OK	
factorial test	
► many_factors2 ✓ OK	
factorial test	
▶ big_powers ✓ OK	
powers of 2 and 3	
► extreme_maximal ✓ OK	
extreme test with maximal values	