Codility_

CodeCheck Report: trainingGGUSM9-M8X

Test Name:

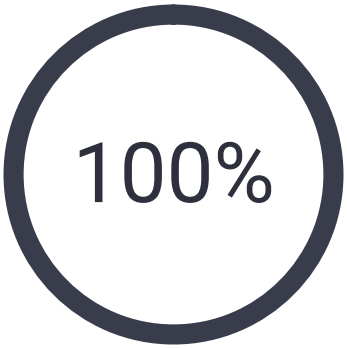
Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
CommonPrimeDivisors Python	10 min	100%

Total score



Tasks Details

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

Write a function:

```
def solution(A, B)
```

Solution

Programming language used:	Python
Total time used:	10 minutes
Effective time used:	10 minutes
Notes:	*not defined yet*

Task timeline

15:27:36 15:37:31

Code: 15:37:31 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debugging purpo
```

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].


```python
# print("this is a debug message")

def solution(A, B):
    # Implement your solution here
    # pass
    def gcd(a, b):
        if b == 0:
            return a
        return gcd(b, a % b)

    def has_same_prime_divisors(a, b):
        gcd_value = gcd(a, b)

        while a != 1:
            a_gcd = gcd(a, gcd_value)
            if a_gcd == 1:
                break
            a //= a_gcd

        if a != 1:
            return False

        while b != 1:
            b_gcd = gcd(b, gcd_value)
            if b_gcd == 1:
                break
            b //= b_gcd

        return b == 1

    count = 0

    for i in range(len(A)):
        if has_same_prime_divisors(A[i], B[i]
            count += 1

    return count


```

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	✔ OK

powers of primes		
▶	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