Open Feedback Dialog

codility

Training center

Check out Codility training tasks

Demo ticket

Session

ID: demoNGU35H-GX2
Time limit: 120 min.

Status: closed

Created on: 2014-03-16 07:49 UTC
Started on: 2014-03-16 07:49 UTC
Finished on: 2014-03-16 07:57 UTC

Tasks in test

Task score

Test score

100%

100 out of 100 points

MEDIUM

1. CommonPrimeDivisors

Check whether two numbers have the same prime divisors.

score: 100 of 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)
```

that, given two non-empty zero-indexed 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.
Assume that:

- Z is an integer within the range [1..6,000];
- each element of arrays A, B is an integer within the range [1..2147483647].

Complexity:

- expected worst-case time complexity is O(Z*log(max(A)+max(B))^2);

Solution

Programming language used: Python

Total time used: 9 minutes

Effective time used: 9 minutes

Notes: correct functionality and scalability

Task timeline

07:49:21 07:57:52

Code: 07:57:52 UTC, py, final, score: **100.00**

```
01. def solution(A, B):
02.     count = 0
03.     size = len(A)
04.     for i in xrange(size):
05.         a = A[i]
06.         b = B[i]
07.         d = c = gcd(a, b)
08.         while a != 1 and d != 1:
09.             a /= d
10.             d = gcd(a, c)
11.         d = c
12.         while b != 1 and d != 1:
13.             b /= d
14.             d = gcd(b, c)
15.         if a == b == 1:
16.             count += 1
17.     return count
18.
19. def gcd(a, b):
20.     return b if a % b == 0 else gcd(b, a % b)
```

Analysis

O(Z*log(max(A)+max(B))^2);
- expected worst-case space complexity is O(1), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Detected time complexity:
O(Z * log(max(A) + max(B))**2)

test	time	result
example example test	0.050 s.	**OK**
extreme extreme test with small values	0.050 s.	**OK**
simple_1 simple test with small values	0.050 s.	**OK**
simple_2 simple test with small values	0.050 s.	**OK**
primes powers of primes	0.050 s.	**OK**
small_primes small primes	0.050 s.	**OK**
small_all_pairs all pairs 1-10, length = 100	0.050 s.	**OK**
small_random small random test, length = 100	0.050 s.	**OK**
large_all_pairs all pairs 1-70, length = ~5,000	0.080 s.	**OK**
large_random large random tests, length = ~6,000	0.110 s.	**OK**
many_factors factorial test	0.110 s.	**OK**
many_factors2 factorial test	0.100 s.	**OK**
big_powers powers of 2 and 3	0.090 s.	**OK**
extreme_maximal extreme test with maximal values	0.080 s.	**OK**

Training center