Open Feedback Dialog

codility

Training center

Check out Codility training tasks

Demo ticket

Session

ID: demoN8T32H-67U
Time limit: 120 min.

Status: closed

Created on: 2014-03-20 16:16 UTC
Started on: 2014-03-20 16:16 UTC
Finished on: 2014-03-20 16:56 UTC

Tasks in test

1 | {} ~~CountSemiprimes~~

Task score

100%

Test score

?

100%

100 out of 100 points

MEDIUM

1. CountSemiprimes

Count the semiprime numbers in the given range [a..b]

score: 100 of 100

Task description

The *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.
The *semiprime* is a natural number that is the product of two (not necessarily distinct) prime numbers. The first few semiprimes are 4, 6, 9, 10, 14, 15, 21, 22, 25, 26.
You are given two non-empty zero-indexed arrays P and Q, each consisting of M integers. These arrays represent queries about the number of semiprimes within specified ranges.
Query K requires you to find the number of semiprimes within the range (P[K], Q[K]), where 1 ≤ P[K] ≤ Q[K] ≤ N.
For example, consider an integer N = 26 and arrays P, Q such that:

```
P[0] = 1    Q[0] = 26
P[1] = 4    Q[1] = 10
P[2] = 16   Q[2] = 20
```

The number of semiprimes within each of these ranges is as follows:

- (1, 26) is 10,
- (4, 10) is 4,
- (16, 20) is 0.

Write a function:

```
def solution(N, P, Q)
```

that, given an integer N and two non-empty zero-indexed arrays P and Q consisting of M integers, returns an array consisting of M elements specifying the consecutive answers to all the queries.
For example, given an integer N = 26 and arrays P, Q such that:

```
P[0] = 1    Q[0] = 26
P[1] = 4    Q[1] = 10
P[2] = 16   Q[2] = 20
```

the function should return the values [10, 4, 0], as explained above.
Assume that:

- N is an integer within the range [1..50,000];
- M is an integer within the range [1..30,000];
- each element of array P is an integer within the range

Solution

Programming language used: Python

Total time used: 40 minutes ?

Effective time used: 40 minutes ?

Notes: correct functionality and scalability

Task timeline ?

16:16:59 16:56:29

Code: 16:56:29 UTC, py, final, score: **100.00**

```python
def solution(N, P, Q):
    F = [0] * (N + 1)
    R = range(N+1)
    i = 2
    while i <= N:
        if F[i] == 0:
            j = i
            while j <= N:
                F[j] += 1
                R[j] /= i
                j += i
        i += 1

    count = [0] * (N + 1)
    for i in xrange(4, N+1):
        count[i] = count[i-1]
        if (F[i] == 1 and i == R[i] * R[i])\
            or (F[i] == 2 and R[i] == 1):
            count[i] += 1

    M = len(P)
    ret = [0] * M
```

- each element of array P is an integer within the range [1..N];
- each element of array Q is an integer within the range [1..N];
- P[i] ≤ Q[i].

Complexity:

- expected worst-case time complexity is O(N*log(log(N))+M);
- expected worst-case space complexity is O(N+M), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

```
    for i in xrange(M):
        ret[i] = count[Q[i]] - count[P[i]-1]
    return ret

```

Analysis

Detected time complexity:

O(N * log(log(N)) + M)

test	time	result
example example test	0.050 s.	**OK**
extreme_one small N = 1	0.050 s.	**OK**
extreme_four small N = 4	0.050 s.	**OK**
small_functional small functional	0.050 s.	**OK**
small_random small random, length = ~40	0.050 s.	**OK**
medium_random small random, length = ~300	0.050 s.	**OK**
large_small_slices large with very small slices, length = ~30,000	0.260 s.	**OK**
large_random1 large random, length = ~30,000	0.320 s.	**OK**
large_random2 large random, length = ~30,000	0.310 s.	**OK**
extreme_large all max ranges	0.300 s.	**OK**

Training center