Codility_

CodeCheck Report: training353P9U-DG6

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
CountSemiprimes Python ⚠	9 min	100%

Total score

100%

Tasks Details

		Task Score	Correctness	Performance
Medium	1. CountSemiprimes Count the semiprime numbers in the given range [a..b]	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 *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 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.

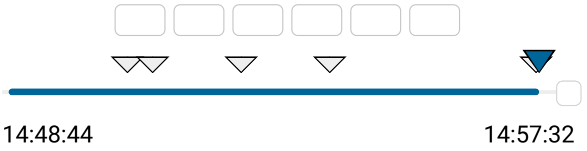
For example, consider an integer N = 26 and arrays P, Q such that:

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

Solution

Programming language used:	Python	
Total time used:	9 minutes	?
Effective time used:	9 minutes	?
Notes:	*not defined yet*	

Task timeline ?



Code: 14:57:32 UTC, py, show code in pop-up

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 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.

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

- N is an integer within the range [1..50,000];
- M is an integer within the range [1..30,000];
- each element of arrays P and Q is an integer within the range [1..N];
- P[i] ≤ Q[i].

final, score: **100**

```
# you can write to stdout for debugging pu
# print("this is a debug message")
from math import sqrt

def solution(N, P, Q):
    # Implement your solution here
    # pass
    sieve = [0] * (N + 1)
    prefix_sum = [0] * (N + 1)
    semiprime_count = 0

    # Generate semiprime counts
    for i in range(2, int(sqrt(N))  + 1):
        if sieve[i] == 0:
            for j in range(i * i, N + 1, i
                if sieve[j] == 0:
                    sieve[j] = i

    for i in range(2, N + 1):
        if sieve[i] != 0 and sieve[i // si
            semiprime_count += 1
        prefix_sum[i] = semiprime_count

    result = []
    for i in range(len(P)):
        count = prefix_sum[Q[i]] - prefix_
        result.append(count)

    return result
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N * log(log(N)) + M)**

expand all **Example tests**

▶ example ✔ OK
example test

expand all **Correctness tests**

▶ extreme_one ✔ OK
small N = 1

▶ extreme_four ✔ OK
small N = 4

▶ small_functional ✔ OK
small functional

▶ small_random ✔ OK
small random, length = ~40

expand all **Performance tests**

▶ medium_random ✔ OK
small random, length = ~300

▶	large_small_slices large with very small slices, length = ~30,000	✔ OK
▶	large_random1 large random, length = ~30,000	✔ OK
▶	large_random2 large random, length = ~30,000	✔ OK
▶	extreme_large all max ranges	✔ OK