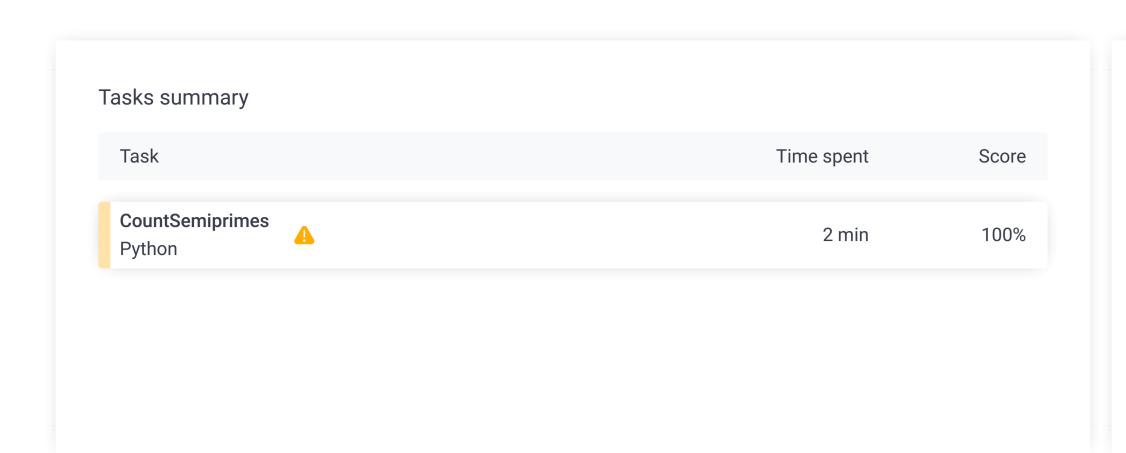
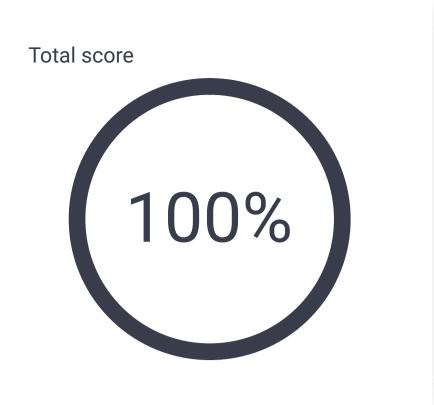
Test Name:

Timeline Summary





## **Tasks Details**



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

Performance **Task Score** Correctness 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  $1 \le P[K] \le Q[K] \le N$ .

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

Q[0] = 26P[0] = 1P[1] = 4Q[1] = 10P[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 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:

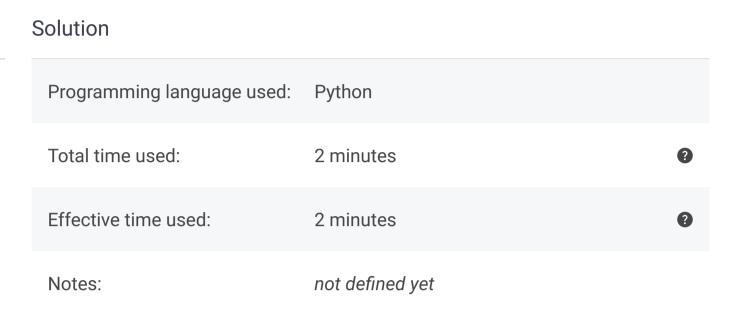
P[0] = 1Q[0] = 26P[1] = 4Q[1] = 10P[2] = 16 Q[2] = 20

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

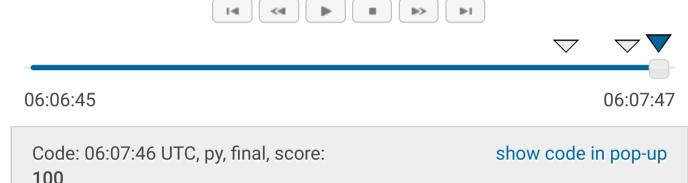
Write an **efficient** algorithm for the following assumptions:

- 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];
- $P[i] \leq Q[i]$ .

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



Task timeline



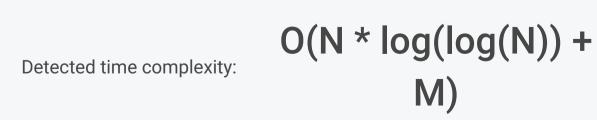
```
100
    # you can write to stdout for debugging purposes, e.g.
    # print("this is a debug message")
     def solution(N, P, Q):
        # write your code in Python 3.
        M = len(P)
        if M == 0:
            return []
        arr = [1] * (N+1)
10
        i = 1
11
        while True:
12
            i += 1
13
            k = i
14
            while True:
15
                 k = k + i
16
                 if k <= N:
                    arr[k] = 0
17
18
                 else:
19
                     break
20
            if i > N//2+1:
21
                 break
22
23
        arr1 = [0] * (N+1)
        for i in range(2, N//2+1):
24
25
            for j in range(i, N//i+1):
26
                if i*j <= N:
27
                     if arr[i] and arr[j]:
28
                         arr1[i*j] = 1
29
30
        cumulative = [0] * (N+2)
31
        for k in range(2, N+2):
32
            if k < N+1:
33
                 if arr1[k]:
34
                     cumulative[k] = cumulative[k-1] + 1
35
                 else:
36
                     cumulative[k] = cumulative[k-1]
37
            else:
                 cumulative[k] = cumulative[k-1]
38
39
40
        result = []
41
        for l in range(M):
42
            result.append(cumulative[Q[l]]-cumulative[P[l]-1])
43
44
        return result
```

## Analysis summary

all max ranges

The solution obtained perfect score.

## Analysis



		••••
expand	d all Examp	ole tests
•	example example test	✓ OK
expand	d all Correctn	iess tests
•	extreme_one small N = 1	✓ OK
•	extreme_four small N = 4	✓ OK
•	small_functional small functional	<b>✓</b> OK
•	small_random small random, length = ~40	<b>✓</b> OK
expand	l all Performa	ance tests
•	medium_random small random, length = ~300	✓ OK
•	large_small_slices large with very small slices, length = ~30,00	<b>✓ OK</b> 00
•	large_random1 large random, length = ~30,000	<b>✓</b> OK
•	large_random2 large random, length = ~30,000	<b>✓</b> OK
•	extreme_large	<b>✓</b> OK