PAINLESS

CountSemiprimes

START

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

Programming language: C++

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 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 $1 \leq P[K] \leq Q[K] \leq N$.

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:

```
vector<int> solution(int N, vector<int> &P,
vector<int> &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 arrays P, 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.

Future training