Count Prime Number in Range LeetCode

You are given a positive integer 'n'. Write a program to find and count all prime numbers up to 'n' using two different approaches: an optimized brute force method and the Sieve of Eratosthenes algorithm. Display the prime numbers and return the count for each approach.

Approach 1: Check if a number is prime using brute force

- This function takes an integer 'n' as input and determines whether 'n' is prime using a brute force approach.
- It iterates from 2 to 'n-1' and checks if any number divides 'n' evenly.
- **Time Complexity: O(n)**
- **Space Complexity: O(1)**

Approach 2: Check if a number is prime using optimized Brute Force approach

- This function takes an integer 'n' as input and determines whether 'n' is prime using an optimized approach.
- It iterates from 2 to the square root of 'n' and checks if any number divides 'n' evenly.
- **Time Complexity: O(sqrt(n))**
- **Space Complexity: O(1)**

Approach 3: Count the prime numbers up to n using Sieve of Eratosthenes

- This function counts and displays the prime numbers up to 'n' using the Sieve of Eratosthenes algorithm.
- It creates a boolean vector **isPrime** of size 'n' to mark numbers as prime or non-prime.
- It iterates from 2 to 'n-1' and marks all multiples of each prime number as non-prime.
- It counts the prime numbers and returns the count.
- **Time Complexity: O(n * log(log(n)))**
- Initializing the isPrime vector: This step takes O(n) time since we create a boolean vector of size 'n' and initialize all elements to true.
- Sieving the non-prime numbers: In the outer loop, we iterate from 2 to 'n-1', and for each number 'i', we check if it is marked as prime in the isPrime vector. If it is prime,

we increment the count, and then we mark all multiples of 'i' as non-prime. The inner loop runs approximately n/i times for each value of 'i'. Therefore, the time complexity of sieving the non-prime numbers is roughly O(n/2 + n/3 + n/4 + ... + n/n), which is asymptotically equivalent to O(n * log(log(n))). This step is the most time-consuming part of the Sieve of Eratosthenes algorithm.

- Overall, the time complexity of the countPrimesSieveOfEratosthenes function is O(n * log(log(n))).
- **Space Complexity: O(n)**