Problem 204: Count Primes

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Given an integer

n

, return

the number of prime numbers that are strictly less than

n

.

Example 1:

Input:

n = 10

Output:

4

Explanation:

There are 4 prime numbers less than 10, they are 2, 3, 5, 7.

Example 2:

Input:

n = 0

Output:

0

Example 3:

Input:

n = 1

Output:

0

Constraints:

0 <= n <= 5 * 10

6

Code Snippets

C++:

```
class Solution {
public:
int countPrimes(int n) {

}
};
```

Java:

```
class Solution {
public int countPrimes(int n) {

}
}
```

Python3:

```
class Solution:
def countPrimes(self, n: int) -> int:
```

Python:

```
class Solution(object):
def countPrimes(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {number}
*/
var countPrimes = function(n) {

};
```

TypeScript:

```
function countPrimes(n: number): number {

};
```

C#:

```
public class Solution {
public int CountPrimes(int n) {

}
}
```

C:

```
int countPrimes(int n) {

}
```

Go:

```
func countPrimes(n int) int {

}
```

Kotlin:

```
class Solution {
fun countPrimes(n: Int): Int {

}
}
```

Swift:

```
class Solution {
func countPrimes(_ n: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_primes(n: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @return {Integer}
def count_primes(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function countPrimes($n) {

}
}
```

Dart:

```
class Solution {
int countPrimes(int n) {

}
}
```

Scala:

```
object Solution {
def countPrimes(n: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_primes(n :: integer) :: integer
def count_primes(n) do

end
end
```

Erlang:

```
-spec count_primes(N :: integer()) -> integer().
count_primes(N) ->
.
```

Racket:

```
(define/contract (count-primes n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Primes
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int countPrimes(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Count Primes
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int countPrimes(int n) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Count Primes
Difficulty: Medium
Tags: array, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def countPrimes(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countPrimes(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count Primes
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
/**
* @param {number} n
* @return {number}
*/
var countPrimes = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Primes
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function countPrimes(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Count Primes
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int CountPrimes(int n) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Count Primes
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int countPrimes(int n) {

}
```

Go Solution:

```
// Problem: Count Primes
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func countPrimes(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countPrimes(n: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func countPrimes(_ n: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Primes
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn count_primes(n: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Integer}
def count_primes(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function countPrimes($n) {

}
}
```

Dart Solution:

```
class Solution {
int countPrimes(int n) {

}
}
```

Scala Solution:

```
object Solution {
def countPrimes(n: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_primes(n :: integer) :: integer
def count_primes(n) do

end
end
```

Erlang Solution:

```
-spec count_primes(N :: integer()) -> integer().
count_primes(N) ->
.
```

Racket Solution:

```
(define/contract (count-primes n)
(-> exact-integer? exact-integer?)
)
```