

Problem 762: Prime Number of Set Bits in Binary Representation

Problem Information

Difficulty: [Easy](#)

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given two integers

left

and

right

, return

the

count

of numbers in the

inclusive

range

[left, right]

having a

prime number of set bits

in their binary representation

Recall that the

number of set bits

an integer has is the number of

1

's present when written in binary.

For example,

21

written in binary is

10101

, which has

3

set bits.

Example 1:

Input:

left = 6, right = 10

Output:

4

Explanation:

6 -> 110 (2 set bits, 2 is prime) 7 -> 111 (3 set bits, 3 is prime) 8 -> 1000 (1 set bit, 1 is not prime) 9 -> 1001 (2 set bits, 2 is prime) 10 -> 1010 (2 set bits, 2 is prime) 4 numbers have a prime number of set bits.

Example 2:

Input:

left = 10, right = 15

Output:

5

Explanation:

10 -> 1010 (2 set bits, 2 is prime) 11 -> 1011 (3 set bits, 3 is prime) 12 -> 1100 (2 set bits, 2 is prime) 13 -> 1101 (3 set bits, 3 is prime) 14 -> 1110 (3 set bits, 3 is prime) 15 -> 1111 (4 set bits, 4 is not prime) 5 numbers have a prime number of set bits.

Constraints:

$1 \leq left \leq right \leq 10$

6

$0 \leq right - left \leq 10$

4

Code Snippets

C++:

```
class Solution {  
public:  
    int countPrimeSetBits(int left, int right) {
```

```
    }
};
```

Java:

```
class Solution {
    public int countPrimeSetBits(int left, int right) {
        return 0;
    }
}
```

Python3:

```
class Solution:
    def countPrimeSetBits(self, left: int, right: int) -> int:
```

Python:

```
class Solution(object):
    def countPrimeSetBits(self, left, right):
        """
        :type left: int
        :type right: int
        :rtype: int
        """

```

JavaScript:

```
/**
 * @param {number} left
 * @param {number} right
 * @return {number}
 */
var countPrimeSetBits = function(left, right) {
    ;
};
```

TypeScript:

```
function countPrimeSetBits(left: number, right: number): number {
```

```
};
```

C#:

```
public class Solution {  
    public int CountPrimeSetBits(int left, int right) {  
        }  
    }
```

C:

```
int countPrimeSetBits(int left, int right) {  
}
```

Go:

```
func countPrimeSetBits(left int, right int) int {  
}
```

Kotlin:

```
class Solution {  
    fun countPrimeSetBits(left: Int, right: Int): Int {  
        }  
    }
```

Swift:

```
class Solution {  
    func countPrimeSetBits(_ left: Int, _ right: Int) -> Int {  
        }  
    }
```

Rust:

```
impl Solution {  
    pub fn count_prime_set_bits(left: i32, right: i32) -> i32 {
```

```
}
```

```
}
```

Ruby:

```
# @param {Integer} left
# @param {Integer} right
# @return {Integer}
def count_prime_set_bits(left, right)

end
```

PHP:

```
class Solution {

    /**
     * @param Integer $left
     * @param Integer $right
     * @return Integer
     */
    function countPrimeSetBits($left, $right) {

    }
}
```

Dart:

```
class Solution {
int countPrimeSetBits(int left, int right) {

}
```

Scala:

```
object Solution {
def countPrimeSetBits(left: Int, right: Int): Int = {

}
```

Elixir:

```
defmodule Solution do
@spec count_prime_set_bits(left :: integer, right :: integer) :: integer
def count_prime_set_bits(left, right) do

end
end
```

Erlang:

```
-spec count_prime_set_bits(Left :: integer(), Right :: integer()) ->
integer().
count_prime_set_bits(Left, Right) ->
.
```

Racket:

```
(define/contract (count-prime-set-bits left right)
(-> exact-integer? exact-integer? exact-integer?))
```

Solutions

C++ Solution:

```
/*
 * Problem: Prime Number of Set Bits in Binary Representation
 * Difficulty: Easy
 * Tags: math
 *
 * Approach: Optimized algorithm based on problem constraints
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
public:
    int countPrimeSetBits(int left, int right) {
}
```

```
};
```

Java Solution:

```
/**  
 * Problem: Prime Number of Set Bits in Binary Representation  
 * Difficulty: Easy  
 * Tags: math  
 *  
 * Approach: Optimized algorithm based on problem constraints  
 * Time Complexity: O(n) to O(n^2) depending on approach  
 * Space Complexity: O(1) to O(n) depending on approach  
 */  
  
class Solution {  
    public int countPrimeSetBits(int left, int right) {  
  
    }  
}
```

Python3 Solution:

```
"""  
Problem: Prime Number of Set Bits in Binary Representation  
Difficulty: Easy  
Tags: math  
  
Approach: Optimized algorithm based on problem constraints  
Time Complexity: O(n) to O(n^2) depending on approach  
Space Complexity: O(1) to O(n) depending on approach  
"""  
  
class Solution:  
    def countPrimeSetBits(self, left: int, right: int) -> int:  
        # TODO: Implement optimized solution  
        pass
```

Python Solution:

```
class Solution(object):  
    def countPrimeSetBits(self, left, right):
```

```
"""
:type left: int
:type right: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
 * Problem: Prime Number of Set Bits in Binary Representation
 * Difficulty: Easy
 * Tags: math
 *
 * Approach: Optimized algorithm based on problem constraints
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * @param {number} left
 * @param {number} right
 * @return {number}
 */
var countPrimeSetBits = function(left, right) {

};
```

TypeScript Solution:

```
/**
 * Problem: Prime Number of Set Bits in Binary Representation
 * Difficulty: Easy
 * Tags: math
 *
 * Approach: Optimized algorithm based on problem constraints
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

function countPrimeSetBits(left: number, right: number): number {
```

```
};
```

C# Solution:

```
/*
 * Problem: Prime Number of Set Bits in Binary Representation
 * Difficulty: Easy
 * Tags: math
 *
 * Approach: Optimized algorithm based on problem constraints
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

public class Solution {
    public int CountPrimeSetBits(int left, int right) {

    }
}
```

C Solution:

```
/*
 * Problem: Prime Number of Set Bits in Binary Representation
 * Difficulty: Easy
 * Tags: math
 *
 * Approach: Optimized algorithm based on problem constraints
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

int countPrimeSetBits(int left, int right) {

}
```

Go Solution:

```
// Problem: Prime Number of Set Bits in Binary Representation
// Difficulty: Easy
// Tags: math
```

```

// 
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func countPrimeSetBits(left int, right int) int {

}

```

Kotlin Solution:

```

class Solution {
    fun countPrimeSetBits(left: Int, right: Int): Int {
        return 0
    }
}

```

Swift Solution:

```

class Solution {
    func countPrimeSetBits(_ left: Int, _ right: Int) -> Int {
        return 0
    }
}

```

Rust Solution:

```

// Problem: Prime Number of Set Bits in Binary Representation
// Difficulty: Easy
// Tags: math
// 

// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
    pub fn count_prime_set_bits(left: i32, right: i32) -> i32 {
        return 0
    }
}

```

Ruby Solution:

```
# @param {Integer} left
# @param {Integer} right
# @return {Integer}
def count_prime_set_bits(left, right)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param Integer $left
     * @param Integer $right
     * @return Integer
     */
    function countPrimeSetBits($left, $right) {

    }
}
```

Dart Solution:

```
class Solution {
    int countPrimeSetBits(int left, int right) {
    }
}
```

Scala Solution:

```
object Solution {
    def countPrimeSetBits(left: Int, right: Int): Int = {
    }
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_prime_set_bits(left :: integer, right :: integer) :: integer
def count_prime_set_bits(left, right) do

end
end
```

Erlang Solution:

```
-spec count_prime_set_bits(Left :: integer(), Right :: integer()) ->
integer().
count_prime_set_bits(Left, Right) ->
.
```

Racket Solution:

```
(define/contract (count-prime-set-bits left right)
(-> exact-integer? exact-integer? exact-integer?))
```