Problem 3610: Minimum Number of Primes to Sum to Target

Problem Information

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

Problem Description

You are given two integers

n

and

m

.

You have to select a multiset of

prime numbers

from the

first

m

prime numbers such that the sum of the selected primes is

exactly

n

. You may use each prime number

multiple

times.

Return the

minimum

number of prime numbers needed to sum up to

n

, or -1 if it is not possible.

Example 1:

Input:

n = 10, m = 2

Output:

4

Explanation:

The first 2 primes are [2, 3]. The sum 10 can be formed as 2 + 2 + 3 + 3, requiring 4 primes.

Example 2:

Input:

n = 15, m = 5

Output:

3

Explanation:

The first 5 primes are [2, 3, 5, 7, 11]. The sum 15 can be formed as 5 + 5 + 5, requiring 3 primes.

Example 3:

Input:

n = 7, m = 6

Output:

1

Explanation:

The first 6 primes are [2, 3, 5, 7, 11, 13]. The sum 7 can be formed directly by prime 7, requiring only 1 prime.

Constraints:

1 <= n <= 1000

1 <= m <= 1000

Code Snippets

C++:

```
class Solution {
public:
int minNumberOfPrimes(int n, int m) {

}
};
```

Java:

```
class Solution {
public int minNumberOfPrimes(int n, int m) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number} n
* @param {number} m
* @return {number}
*/
var minNumberOfPrimes = function(n, m) {

};
```

TypeScript:

```
function minNumberOfPrimes(n: number, m: number): number {

};
```

C#:

```
public class Solution {
public int MinNumberOfPrimes(int n, int m) {

}
}
```

C:

```
int minNumberOfPrimes(int n, int m) {

}
```

Go:

```
func minNumberOfPrimes(n int, m int) int {

}
```

Kotlin:

```
class Solution {
fun minNumberOfPrimes(n: Int, m: Int): Int {

}
}
```

Swift:

```
class Solution {
func minNumberOfPrimes(_ n: Int, _ m: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_number_of_primes(n: i32, m: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} m
# @return {Integer}
def min_number_of_primes(n, m)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $m
* @return Integer
*/
function minNumberOfPrimes($n, $m) {

}
}
```

Dart:

```
class Solution {
int minNumberOfPrimes(int n, int m) {

}
}
```

Scala:

```
object Solution {
def minNumberOfPrimes(n: Int, m: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_number_of_primes(n :: integer, m :: integer) :: integer
def min_number_of_primes(n, m) do
```

```
end
end
```

Erlang:

```
-spec min_number_of_primes(N :: integer(), M :: integer()) -> integer().
min_number_of_primes(N, M) ->
.
```

Racket:

```
(define/contract (min-number-of-primes n m)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Number of Primes to Sum to Target
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int minNumberOfPrimes(int n, int m) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Primes to Sum to Target
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int minNumberOfPrimes(int n, int m) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Primes to Sum to Target
Difficulty: Medium
Tags: array, dp, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Number of Primes to Sum to Target
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number} n
* @param {number} m
* @return {number}
*/
var minNumberOfPrimes = function(n, m) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Primes to Sum to Target
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function minNumberOfPrimes(n: number, m: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Number of Primes to Sum to Target
* Difficulty: Medium
* Tags: array, dp, math
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int MinNumberOfPrimes(int n, int m) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Primes to Sum to Target
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int minNumberOfPrimes(int n, int m) {

}
```

Go Solution:

```
// Problem: Minimum Number of Primes to Sum to Target
// Difficulty: Medium
// Tags: array, dp, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func minNumberOfPrimes(n int, m int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minNumberOfPrimes(n: Int, m: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func minNumberOfPrimes(_ n: Int, _ m: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Number of Primes to Sum to Target
// Difficulty: Medium
// Tags: array, dp, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn min_number_of_primes(n: i32, m: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} m
# @return {Integer}
def min_number_of_primes(n, m)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $m
* @return Integer
*/
function minNumberOfPrimes($n, $m) {

}
}
```

Dart Solution:

```
class Solution {
int minNumberOfPrimes(int n, int m) {

}
}
```

Scala Solution:

```
object Solution {
def minNumberOfPrimes(n: Int, m: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_number_of_primes(n :: integer, m :: integer) :: integer
def min_number_of_primes(n, m) do

end
end
```

Erlang Solution:

```
-spec min_number_of_primes(N :: integer(), M :: integer()) -> integer().
min_number_of_primes(N, M) ->
.
```

Racket Solution:

```
(define/contract (min-number-of-primes n m)
(-> exact-integer? exact-integer? exact-integer?)
)
```