# Problem 2719: Count of Integers

## Problem Information

**Difficulty:** Hard
**Acceptance Rate:** 0.00%
**Paid Only:** No

## Problem Description

You are given two numeric strings

num1

and

num2

and two integers

max_sum

and

min_sum

. We denote an integer

x

to be

good

if:

num1 <= x <= num2

min_sum <= digit_sum(x) <= max_sum

.

Return

the number of good integers

. Since the answer may be large, return it modulo

10

9

+ 7

.

Note that

digit_sum(x)

denotes the sum of the digits of

x

.

Example 1:

Input:

num1 = "1", num2 = "12",

min_sum

= 1, max_sum = 8

Output:

11

Explanation:

There are 11 integers whose sum of digits lies between 1 and 8 are 1,2,3,4,5,6,7,8,10,11, and 12. Thus, we return 11.

Example 2:

Input:

num1 = "1", num2 = "5",

min_sum

= 1, max_sum = 5

Output:

5

Explanation:

The 5 integers whose sum of digits lies between 1 and 5 are 1,2,3,4, and 5. Thus, we return 5.

Constraints:

1 <= num1 <= num2 <= 10

22

1 <= min_sum <= max_sum <= 400

## Code Snippets

**C++:**

```
class Solution {
public:
int count(string num1, string num2, int min_sum, int max_sum) {

}
};
```

**Java:**

```
class Solution {
public int count(String num1, String num2, int min_sum, int max_sum) {

}
}
```

**Python3:**

```
class Solution:
def count(self, num1: str, num2: str, min_sum: int, max_sum: int) -> int:
```

**Python:**

```
class Solution(object):
def count(self, num1, num2, min_sum, max_sum):
"""
:type num1: str
:type num2: str
:type min_sum: int
:type max_sum: int
:rtype: int
"""
```

**JavaScript:**

```
/**
* @param {string} num1
* @param {string} num2
* @param {number} min_sum
* @param {number} max_sum
* @return {number}
*/
```

```
var count = function(num1, num2, min_sum, max_sum) {

};
```

**TypeScript:**

```
function count(num1: string, num2: string, min_sum: number, max_sum: number):
number {

};
```

**C#:**

```
public class Solution {
public int Count(string num1, string num2, int min_sum, int max_sum) {

}
}
```

**C:**

```
int count(char* num1, char* num2, int min_sum, int max_sum) {

}
```

**Go:**

```
func count(num1 string, num2 string, min_sum int, max_sum int) int {

}
```

**Kotlin:**

```
class Solution {
fun count(num1: String, num2: String, min_sum: Int, max_sum: Int): Int {

}
}
```

**Swift:**

```
class Solution {
func count(_ num1: String, _ num2: String, _ min_sum: Int, _ max_sum: Int) ->
Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn count(num1: String, num2: String, min_sum: i32, max_sum: i32) -> i32 {

}
}
```

**Ruby:**

```
# @param {String} num1
# @param {String} num2
# @param {Integer} min_sum
# @param {Integer} max_sum
# @return {Integer}
def count(num1, num2, min_sum, max_sum)

end
```

**PHP:**

```
class Solution {

/**
* @param String $num1
* @param String $num2
* @param Integer $min_sum
* @param Integer $max_sum
* @return Integer
*/
function count($num1, $num2, $min_sum, $max_sum) {

}
}
```

**Dart:**

```
class Solution {
int count(String num1, String num2, int min_sum, int max_sum) {

}
}
```

### Scala:

```
object Solution {
def count(num1: String, num2: String, min_sum: Int, max_sum: Int): Int = {

}
}
```

### Elixir:

```
defmodule Solution do
@spec count(num1 :: String.t, num2 :: String.t, min_sum :: integer, max_sum
:: integer) :: integer
def count(num1, num2, min_sum, max_sum) do

end
end
```

### Erlang:

```
-spec count(Num1 :: unicode:unicode_binary(), Num2 ::
unicode:unicode_binary(), Min_sum :: integer(), Max_sum :: integer()) ->
integer().
count(Num1, Num2, Min_sum, Max_sum) ->
.
```

### Racket:

```
(define/contract (count num1 num2 min_sum max_sum)
(-> string? string? exact-integer? exact-integer? exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Count of Integers
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int count(string num1, string num2, int min_sum, int max_sum) {

}
};
```

**Java Solution:**

```
/**
* Problem: Count of Integers
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int count(String num1, String num2, int min_sum, int max_sum) {

}
}
```

**Python3 Solution:**

```
"""
Problem: Count of Integers
Difficulty: Hard
Tags: string, dp, math
```

```
Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def count(self, num1: str, num2: str, min_sum: int, max_sum: int) -> int:
# TODO: Implement optimized solution
pass
```

**Python Solution:**

```
class Solution(object):
def count(self, num1, num2, min_sum, max_sum):
"""
:type num1: str
:type num2: str
:type min_sum: int
:type max_sum: int
:rtype: int
"""
```

**JavaScript Solution:**

```
/**
* Problem: Count of Integers
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} num1
* @param {string} num2
* @param {number} min_sum
* @param {number} max_sum
* @return {number}
*/
```

```
var count = function(num1, num2, min_sum, max_sum) {

};
```

### TypeScript Solution:

```
/**
* Problem: Count of Integers
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function count(num1: string, num2: string, min_sum: number, max_sum: number):
number {

};
```

### C# Solution:

```
/*
* Problem: Count of Integers
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* 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 Count(string num1, string num2, int min_sum, int max_sum) {

}
}
```

### C Solution:

```
/*
* Problem: Count of Integers
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int count(char* num1, char* num2, int min_sum, int max_sum) {

}
```

**Go Solution:**

```
// Problem: Count of Integers
// Difficulty: Hard
// Tags: string, dp, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func count(num1 string, num2 string, min_sum int, max_sum int) int {

}
```

**Kotlin Solution:**

```
class Solution {
fun count(num1: String, num2: String, min_sum: Int, max_sum: Int): Int {

}
}
```

**Swift Solution:**

```
class Solution {
func count(_ num1: String, _ num2: String, _ min_sum: Int, _ max_sum: Int) ->
Int {


}
}
```

### Rust Solution:

```
// Problem: Count of Integers
// Difficulty: Hard
// Tags: string, dp, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn count(num1: String, num2: String, min_sum: i32, max_sum: i32) -> i32 {

}
}
```

### Ruby Solution:

```
# @param {String} num1
# @param {String} num2
# @param {Integer} min_sum
# @param {Integer} max_sum
# @return {Integer}
def count(num1, num2, min_sum, max_sum)

end
```

### PHP Solution:

```
class Solution {

/**
* @param String $num1
* @param String $num2
* @param Integer $min_sum
* @param Integer $max_sum
* @return Integer
*/
function count($num1, $num2, $min_sum, $max_sum) {

}
}
```

**Dart Solution:**

```
class Solution {
int count(String num1, String num2, int min_sum, int max_sum) {

}
}
```

**Scala Solution:**

```
object Solution {
def count(num1: String, num2: String, min_sum: Int, max_sum: Int): Int = {

}
}
```

**Elixir Solution:**

```
defmodule Solution do
@spec count(num1 :: String.t, num2 :: String.t, min_sum :: integer, max_sum
:: integer) :: integer
def count(num1, num2, min_sum, max_sum) do

end
end
```

**Erlang Solution:**

```
-spec count(Num1 :: unicode:unicode_binary(), Num2 ::
unicode:unicode_binary(), Min_sum :: integer(), Max_sum :: integer()) ->
integer().
count(Num1, Num2, Min_sum, Max_sum) ->
.
```

**Racket Solution:**

```
(define/contract (count num1 num2 min_sum max_sum)
(-> string? string? exact-integer? exact-integer? exact-integer?)
)
```