Problem 2801: Count Stepping Numbers in Range

Problem Information

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

Problem Description

Given two positive integers

low

and

high

represented as strings, find the count of

stepping numbers

in the inclusive range

[low, high]

.

A

stepping number

is an integer such that all of its adjacent digits have an absolute difference of

exactly

1

.

Return

an integer denoting the count of stepping numbers in the inclusive range

[low, high]

.

Since the answer may be very large, return it

modulo

10

9

+ 7

.

Note:

A stepping number should not have a leading zero.

Example 1:

Input:

low = "1", high = "11"

Output:

10

Explanation:

The stepping numbers in the range [1,11] are 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. There are a total of 10 stepping numbers in the range. Hence, the output is 10.

Example 2:

Input:

low = "90", high = "101"

Output:

2

Explanation:

The stepping numbers in the range [90,101] are 98 and 101. There are a total of 2 stepping numbers in the range. Hence, the output is 2.

Constraints:

1 <= int(low) <= int(high) < 10

100

1 <= low.length, high.length <= 100

low

and

high

consist of only digits.

low

and

high

don't have any leading zeros.

Code Snippets

C++:

```
class Solution {
public:
int countSteppingNumbers(string low, string high) {

}
};
```

Java:

```
class Solution {
public int countSteppingNumbers(String low, String high) {

}
}
```

Python3:

```
class Solution:
def countSteppingNumbers(self, low: str, high: str) -> int:
```

Python:

```
class Solution(object):
def countSteppingNumbers(self, low, high):
"""
:type low: str
:type high: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} low
* @param {string} high
* @return {number}
*/
var countSteppingNumbers = function(low, high) {

};
```

TypeScript:

```
function countSteppingNumbers(low: string, high: string): number {

};
```

C#:

```
public class Solution {
public int CountSteppingNumbers(string low, string high) {

}
}
```

C:

```
int countSteppingNumbers(char* low, char* high) {

}
```

Go:

```
func countSteppingNumbers(low string, high string) int {

}
```

Kotlin:

```
class Solution {
fun countSteppingNumbers(low: String, high: String): Int {

}
}
```

Swift:

```
class Solution {
func countSteppingNumbers(_ low: String, _ high: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_stepping_numbers(low: String, high: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} low
# @param {String} high
# @return {Integer}
def count_stepping_numbers(low, high)

end
```

PHP:

```
class Solution {

/**
* @param String $low
* @param String $high
* @return Integer
*/
function countSteppingNumbers($low, $high) {

}
}
```

Dart:

```
class Solution {
int countSteppingNumbers(String low, String high) {
```

```
}
}
```

Scala:

```
object Solution {
def countSteppingNumbers(low: String, high: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_stepping_numbers(low :: String.t, high :: String.t) :: integer
def count_stepping_numbers(low, high) do

end
end
```

Erlang:

```
-spec count_stepping_numbers(Low :: unicode:unicode_binary(), High ::
unicode:unicode_binary()) -> integer().
count_stepping_numbers(Low, High) ->
.
```

Racket:

```
(define/contract (count-stepping-numbers low high)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Stepping Numbers in Range
```

```
* Difficulty: Hard
* Tags: string, dp
*
* 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 countSteppingNumbers(string low, string high) {

}
};
```

Java Solution:

```
/**
* Problem: Count Stepping Numbers in Range
* Difficulty: Hard
* Tags: string, dp
*
* 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 countSteppingNumbers(String low, String high) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Stepping Numbers in Range
Difficulty: Hard
Tags: string, dp

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 countSteppingNumbers(self, low: str, high: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countSteppingNumbers(self, low, high):
"""
:type low: str
:type high: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count Stepping Numbers in Range
* Difficulty: Hard
* Tags: string, dp
*
* 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} low
* @param {string} high
* @return {number}
*/
var countSteppingNumbers = function(low, high) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Stepping Numbers in Range
* Difficulty: Hard
* Tags: string, dp
*
* 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 countSteppingNumbers(low: string, high: string): number {

};
```

C# Solution:

```
/*
* Problem: Count Stepping Numbers in Range
* Difficulty: Hard
* Tags: string, dp
*
* 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 CountSteppingNumbers(string low, string high) {

}
}
```

C Solution:

```
/*
* Problem: Count Stepping Numbers in Range
* Difficulty: Hard
* Tags: string, dp
*
* 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 countSteppingNumbers(char* low, char* high) {

}
```

Go Solution:

```
// Problem: Count Stepping Numbers in Range
// Difficulty: Hard
// Tags: string, dp
//
// 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 countSteppingNumbers(low string, high string) int {

}
```

Kotlin Solution:

```
class Solution {
fun countSteppingNumbers(low: String, high: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func countSteppingNumbers(_ low: String, _ high: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Stepping Numbers in Range
// Difficulty: Hard
// Tags: string, dp
```

```
//
// 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_stepping_numbers(low: String, high: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} low
# @param {String} high
# @return {Integer}
def count_stepping_numbers(low, high)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $low
* @param String $high
* @return Integer
*/
function countSteppingNumbers($low, $high) {

}
}
```

Dart Solution:

```
class Solution {
int countSteppingNumbers(String low, String high) {

}
}
```

Scala Solution:

```
object Solution {
def countSteppingNumbers(low: String, high: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_stepping_numbers(low :: String.t, high :: String.t) :: integer
def count_stepping_numbers(low, high) do

end
end
```

Erlang Solution:

```
-spec count_stepping_numbers(Low :: unicode:unicode_binary(), High ::
unicode:unicode_binary()) -> integer().
count_stepping_numbers(Low, High) ->
.
```

Racket Solution:

```
(define/contract (count-stepping-numbers low high)
(-> string? string? exact-integer?)
)
```