Problem 248: Strobogrammatic Number III

Problem Information

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

Problem Description

Given two strings low and high that represent two integers

low

and

high

where

low <= high

, return

the number of

strobogrammatic numbers

in the range

[low, high]

.

A

strobogrammatic number

is a number that looks the same when rotated

180

degrees (looked at upside down).

Example 1:

Input:

low = "50", high = "100"

Output:

3

Example 2:

Input:

low = "0", high = "0"

Output:

1

Constraints:

1 <= low.length, high.length <= 15

low

and

high

consist of only digits.

low <= high

low

and

high

do not contain any leading zeros except for zero itself.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

```
class Solution(object):
def strobogrammaticInRange(self, low, high):
"""
:type low: str
```

```
:type high: str
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
```

```
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (strobogrammatic-in-range low high)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Strobogrammatic Number III
* Difficulty: Hard
* Tags: array, string
*
* 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 strobogrammaticInRange(string low, string high) {

}
};
```

Java Solution:

```
/**
* Problem: Strobogrammatic Number III
* Difficulty: Hard
* Tags: array, string
*
* 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 strobogrammaticInRange(String low, String high) {

}
}
```

Python3 Solution:

```
"""
Problem: Strobogrammatic Number III
```

```
Difficulty: Hard
Tags: array, string

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Strobogrammatic Number III
* Difficulty: Hard
* Tags: array, string
*
* 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 {string} low
* @param {string} high
* @return {number}
*/
var strobogrammaticInRange = function(low, high) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Strobogrammatic Number III
* Difficulty: Hard
* Tags: array, string
*
* 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 strobogrammaticInRange(low: string, high: string): number {

};
```

C# Solution:

```
/*
* Problem: Strobogrammatic Number III
* Difficulty: Hard
* Tags: array, string
*
* 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 StrobogrammaticInRange(string low, string high) {

}
}
```

C Solution:

```
/*
* Problem: Strobogrammatic Number III
```

```
* Difficulty: Hard
* Tags: array, string
*
* 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 strobogrammaticInRange(char* low, char* high) {

}
```

Go Solution:

```
// Problem: Strobogrammatic Number III
// Difficulty: Hard
// Tags: array, string
//
// 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 strobogrammaticInRange(low string, high string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Strobogrammatic Number III
// Difficulty: Hard
// Tags: array, string
//
// 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 strobogrammatic_in_range(low: String, high: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (strobogrammatic-in-range low high)
(-> string? string? exact-integer?)
)
```