Problem 246: Strobogrammatic Number

Problem Information

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

Problem Description

Given a string

num

which represents an integer, return

true

if

num

is a

strobogrammatic number

.

A

strobogrammatic number

is a number that looks the same when rotated

180

degrees (looked at upside down).

Example 1:

Input:

num = "69"

Output:

true

Example 2:

Input:

num = "88"

Output:

true

Example 3:

Input:

num = "962"

Output:

false

Constraints:

1 <= num.length <= 50

num

consists of only digits.

num

does not contain any leading zeros except for zero itself.

Code Snippets

C++:

```
class Solution {
public:
bool isStrobogrammatic(string num) {

}
};
```

Java:

```
class Solution {
public boolean isStrobogrammatic(String num) {

}
}
```

Python3:

```
class Solution:
def isStrobogrammatic(self, num: str) -> bool:
```

Python:

```
class Solution(object):
def isStrobogrammatic(self, num):
"""
:type num: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} num
* @return {boolean}
*/
var isStrobogrammatic = function(num) {

};
```

TypeScript:

```
function isStrobogrammatic(num: string): boolean {

};
```

C#:

```
public class Solution {
public bool IsStrobogrammatic(string num) {

}
}
```

C:

```
bool isStrobogrammatic(char* num) {

}
```

Go:

```
func isStrobogrammatic(num string) bool {

}
```

Kotlin:

```
class Solution {
fun isStrobogrammatic(num: String): Boolean {

}
}
```

Swift:

```
class Solution {
func isStrobogrammatic(_ num: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_strobogrammatic(num: String) -> bool {

}
}
```

Ruby:

```
# @param {String} num
# @return {Boolean}
def is_strobogrammatic(num)

end
```

PHP:

```
class Solution {

/**
* @param String $num
* @return Boolean
*/
function isStrobogrammatic($num) {

}
}
```

Dart:

```
class Solution {
bool isStrobogrammatic(String num) {

}
}
```

Scala:

```
object Solution {
def isStrobogrammatic(num: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_strobogrammatic(num :: String.t) :: boolean
def is_strobogrammatic(num) do

end
end
```

Erlang:

```
-spec is_strobogrammatic(Num :: unicode:unicode_binary()) -> boolean().
is_strobogrammatic(Num) ->
.
```

Racket:

```
(define/contract (is-strobogrammatic num)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Strobogrammatic Number
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
class Solution {
public:
bool isStrobogrammatic(string num) {

}
};
```

Java Solution:

```
/**
* Problem: Strobogrammatic Number
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public boolean isStrobogrammatic(String num) {

}
}
```

Python3 Solution:

```
"""
Problem: Strobogrammatic Number
Difficulty: Easy
Tags: array, string, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def isStrobogrammatic(self, num: str) -> bool:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def isStrobogrammatic(self, num):
"""
:type num: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Strobogrammatic Number
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} num
* @return {boolean}
*/
var isStrobogrammatic = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Strobogrammatic Number
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

function isStrobogrammatic(num: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Strobogrammatic Number
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public bool IsStrobogrammatic(string num) {

}
}
```

C Solution:

```
/*
* Problem: Strobogrammatic Number
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

bool isStrobogrammatic(char* num) {

}
```

Go Solution:

```
// Problem: Strobogrammatic Number
// Difficulty: Easy
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func isStrobogrammatic(num string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isStrobogrammatic(num: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isStrobogrammatic(_ num: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Strobogrammatic Number
// Difficulty: Easy
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn is_strobogrammatic(num: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} num
# @return {Boolean}
def is_strobogrammatic(num)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $num
* @return Boolean
*/
function isStrobogrammatic($num) {

}
}
```

Dart Solution:

```
class Solution {
bool isStrobogrammatic(String num) {

}
}
```

Scala Solution:

```
object Solution {
def isStrobogrammatic(num: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_strobogrammatic(num :: String.t) :: boolean
def is_strobogrammatic(num) do

end
end
```

Erlang Solution:

```
-spec is_strobogrammatic(Num :: unicode:unicode_binary()) -> boolean().
is_strobogrammatic(Num) ->
.
```

Racket Solution:

```
(define/contract (is-strobogrammatic num)
(-> string? boolean?)
)
```