Problem 2283: Check if Number Has Equal Digit Count and Digit Value

Problem Information

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

Problem Description

You are given a

0-indexed

string

num

of length

n

consisting of digits.

Return

true

if for

every

index

i

in the range

0 <= i < n

, the digit

i

occurs

num[i]

times in

num

, otherwise return

false

.

Example 1:

Input:

num = "1210"

Output:

true

Explanation:

num[0] = '1'. The digit 0 occurs once in num. num[1] = '2'. The digit 1 occurs twice in num. num[2] = '1'. The digit 2 occurs once in num. num[3] = '0'. The digit 3 occurs zero times in num. The condition holds true for every index in "1210", so return true.

Example 2:

Input:

num = "030"

Output:

false

Explanation:

num[0] = '0'. The digit 0 should occur zero times, but actually occurs twice in num. num[1] = '3'. The digit 1 should occur three times, but actually occurs zero times in num. num[2] = '0'. The digit 2 occurs zero times in num. The indices 0 and 1 both violate the condition, so return false.

Constraints:

n == num.length

1 <= n <= 10

num

consists of digits.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
bool digitCount(char* num) {

}
```

Go:

```
func digitCount(num string) bool {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (digit-count num)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if Number Has Equal Digit Count and Digit Value
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
bool digitCount(string num) {

}
};
```

Java Solution:

```
/**
* Problem: Check if Number Has Equal Digit Count and Digit Value
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

```
}
}
```

Python3 Solution:

```
"""
Problem: Check if Number Has Equal Digit Count and Digit Value
Difficulty: Easy
Tags: string, hash

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

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check if Number Has Equal Digit Count and Digit Value
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Check if Number Has Equal Digit Count and Digit Value
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function digitCount(num: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if Number Has Equal Digit Count and Digit Value
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
```

```
}
```

C Solution:

```
/*
* Problem: Check if Number Has Equal Digit Count and Digit Value
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

bool digitCount(char* num) {

}
```

Go Solution:

```
// Problem: Check if Number Has Equal Digit Count and Digit Value
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func digitCount(num string) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Check if Number Has Equal Digit Count and Digit Value
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (digit-count num)
(-> string? boolean?)
)
```