Problem 3340: Check Balanced String

Problem Information

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

Problem Description

You are given a string

num

consisting of only digits. A string of digits is called

balanced

if the sum of the digits at even indices is equal to the sum of digits at odd indices.

Return

true

if

num

is

balanced

, otherwise return

false

.

Example 1:

Input:

num

= "1234"

Output:

false

Explanation:

The sum of digits at even indices is

1 + 3 == 4

, and the sum of digits at odd indices is

2 + 4 == 6

.

Since 4 is not equal to 6,

num

is not balanced.

Example 2:

Input:

num

= "24123"

Output:

true

Explanation:

The sum of digits at even indices is

2 + 1 + 3 == 6

, and the sum of digits at odd indices is

4 + 2 == 6

.

Since both are equal the

num

is balanced.

Constraints:

2 <= num.length <= 100

num

consists of digits only

Code Snippets

C++:

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

```
}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
bool isBalanced(char* num) {

}
```

Go:

```
func isBalanced(num string) bool {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Check Balanced String
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Check Balanced String
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public boolean isBalanced(String num) {

}
}
```

Python3 Solution:

```
"""
Problem: Check Balanced String
Difficulty: Easy
Tags: string

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check Balanced String
* Difficulty: Easy
```

```
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Check Balanced String
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function isBalanced(num: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Check Balanced String
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

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

}
}
```

C Solution:

```
/*
* Problem: Check Balanced String
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

bool isBalanced(char* num) {

}
```

Go Solution:

```
// Problem: Check Balanced String
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func isBalanced(num string) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Check Balanced String
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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