Problem 1941: Check if All Characters Have Equal Number of Occurrences

Problem Information

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

Problem Description

Given a string

s

, return

true

if

s

is a

good

string, or

false

otherwise

.

A string

s

is

good

if

all

the characters that appear in

s

have the

same

number of occurrences (i.e., the same frequency).

Example 1:

Input:

s = "abacbc"

Output:

true

Explanation:

The characters that appear in s are 'a', 'b', and 'c'. All characters occur 2 times in s.

Example 2:

Input:

s = "aaabb"

Output:

false

Explanation:

The characters that appear in s are 'a' and 'b'. 'a' occurs 3 times while 'b' occurs 2 times, which is not the same number of times.

Constraints:

1 <= s.length <= 1000

s

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
bool areOccurrencesEqual(string s) {

}
};
```

Java:

```
class Solution {
public boolean areOccurrencesEqual(String s) {

}
}
```

Python3:

```
class Solution:
def areOccurrencesEqual(self, s: str) -> bool:
```

Python:

```
class Solution(object):
def areOccurrencesEqual(self, s):
"""
:type s: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {boolean}
*/
var areOccurrencesEqual = function(s) {

};
```

TypeScript:

```
function areOccurrencesEqual(s: string): boolean {

};
```

C#:

```
public class Solution {
public bool AreOccurrencesEqual(string s) {

}
}
```

C:

```
bool areOccurrencesEqual(char* s) {

}
```

Go:

```
func areOccurrencesEqual(s string) bool {

}
```

Kotlin:

```
class Solution {
fun areOccurrencesEqual(s: String): Boolean {

}
}
```

Swift:

```
class Solution {
func areOccurrencesEqual(_ s: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn are_occurrences_equal(s: String) -> bool {

}
}
```

Ruby:

```
# @param {String} s
# @return {Boolean}
def are_occurrences_equal(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Boolean
*/
function areOccurrencesEqual($s) {

}
}
```

Dart:

```
class Solution {
bool areOccurrencesEqual(String s) {

}
}
```

Scala:

```
object Solution {
def areOccurrencesEqual(s: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec are_occurrences_equal(s :: String.t) :: boolean
def are_occurrences_equal(s) do

end
end
```

Erlang:

```
-spec are_occurrences_equal(S :: unicode:unicode_binary()) -> boolean().
are_occurrences_equal(S) ->
.
```

Racket:

```
(define/contract (are-occurrences-equal s)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if All Characters Have Equal Number of Occurrences
* 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 areOccurrencesEqual(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Check if All Characters Have Equal Number of Occurrences
* 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 areOccurrencesEqual(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Check if All Characters Have Equal Number of Occurrences
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 areOccurrencesEqual(self, s: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def areOccurrencesEqual(self, s):
"""
:type s: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Check if All Characters Have Equal Number of Occurrences
* 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} s
* @return {boolean}
*/
var areOccurrencesEqual = function(s) {
};
```

TypeScript Solution:

```
/**
* Problem: Check if All Characters Have Equal Number of Occurrences
* 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 areOccurrencesEqual(s: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if All Characters Have Equal Number of Occurrences
* 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 AreOccurrencesEqual(string s) {

}
}
```

C Solution:

```
/*
* Problem: Check if All Characters Have Equal Number of Occurrences
* 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 areOccurrencesEqual(char* s) {

}
```

Go Solution:

```
// Problem: Check if All Characters Have Equal Number of Occurrences
// 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 areOccurrencesEqual(s string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun areOccurrencesEqual(s: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func areOccurrencesEqual(_ s: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Check if All Characters Have Equal Number of Occurrences
// 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 are_occurrences_equal(s: String) -> bool {


}
}
```

Ruby Solution:

```
# @param {String} s
# @return {Boolean}
def are_occurrences_equal(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return Boolean
*/
function areOccurrencesEqual($s) {

}
}
```

Dart Solution:

```
class Solution {
bool areOccurrencesEqual(String s) {
}
}
```

Scala Solution:

```
object Solution {
def areOccurrencesEqual(s: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec are_occurrences_equal(s :: String.t) :: boolean
def are_occurrences_equal(s) do

end
end
```

Erlang Solution:

```
-spec are_occurrences_equal(S :: unicode:unicode_binary()) -> boolean().
are_occurrences_equal(S) ->
.
```

Racket Solution:

```
(define/contract (are-occurrences-equal s)
(-> string? boolean?)
)
```