Problem 1832: Check if the Sentence Is Pangram

Problem Information

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

Problem Description

A

pangram

is a sentence where every letter of the English alphabet appears at least once.

Given a string

sentence

containing only lowercase English letters, return

true

if

sentence

is a

pangram

, or

false

otherwise.

Example 1:

Input:

sentence = "thequickbrownfoxjumpsoverthelazydog"

Output:

true

Explanation:

sentence contains at least one of every letter of the English alphabet.

Example 2:

Input:

sentence = "leetcode"

Output:

false

Constraints:

1 <= sentence.length <= 1000

sentence

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
bool checkIfPangram(string sentence) {

}
};
```

Java:

```
class Solution {
public boolean checkIfPangram(String sentence) {

}
}
```

Python3:

```
class Solution:
def checkIfPangram(self, sentence: str) -> bool:
```

Python:

```
class Solution(object):
def checkIfPangram(self, sentence):
"""
:type sentence: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} sentence
* @return {boolean}
*/
var checkIfPangram = function(sentence) {

};
```

TypeScript:

```
function checkIfPangram(sentence: string): boolean {
```

```
};
```

C#:

```
public class Solution {
public bool CheckIfPangram(string sentence) {

}
}
```

C:

```
bool checkIfPangram(char* sentence) {

}
```

Go:

```
func checkIfPangram(sentence string) bool {

}
```

Kotlin:

```
class Solution {
fun checkIfPangram(sentence: String): Boolean {

}
}
```

Swift:

```
class Solution {
func checkIfPangram(_ sentence: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn check_if_pangram(sentence: String) -> bool {
```

```
}
}
```

Ruby:

```
# @param {String} sentence
# @return {Boolean}
def check_if_pangram(sentence)

end
```

PHP:

```
class Solution {

/**
* @param String $sentence
* @return Boolean
*/
function checkIfPangram($sentence) {

}
}
```

Dart:

```
class Solution {
bool checkIfPangram(String sentence) {

}
}
```

Scala:

```
object Solution {
def checkIfPangram(sentence: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec check_if_pangram(sentence :: String.t) :: boolean
def check_if_pangram(sentence) do

end
end
```

Erlang:

```
-spec check_if_pangram(Sentence :: unicode:unicode_binary()) -> boolean().
check_if_pangram(Sentence) ->
.
```

Racket:

```
(define/contract (check-if-pangram sentence)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if the Sentence Is Pangram
* 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 checkIfPangram(string sentence) {

}
};
```

Java Solution:

```
/**
* Problem: Check if the Sentence Is Pangram
* 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 checkIfPangram(String sentence) {

}
}
```

Python3 Solution:

```
"""
Problem: Check if the Sentence Is Pangram
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 checkIfPangram(self, sentence: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def checkIfPangram(self, sentence):
"""
:type sentence: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Check if the Sentence Is Pangram
* 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} sentence
* @return {boolean}
*/
var checkIfPangram = function(sentence) {

};
```

TypeScript Solution:

```
/**
* Problem: Check if the Sentence Is Pangram
* 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 checkIfPangram(sentence: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if the Sentence Is Pangram
* 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 CheckIfPangram(string sentence) {

}
}
```

C Solution:

```
/*
* Problem: Check if the Sentence Is Pangram
* 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 checkIfPangram(char* sentence) {

}
```

Go Solution:

```
// Problem: Check if the Sentence Is Pangram
// 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 checkIfPangram(sentence string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun checkIfPangram(sentence: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func checkIfPangram(_ sentence: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Check if the Sentence Is Pangram
// 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 check_if_pangram(sentence: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} sentence
# @return {Boolean}
def check_if_pangram(sentence)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $sentence
* @return Boolean
*/
function checkIfPangram($sentence) {

}
}
```

Dart Solution:

```
class Solution {
bool checkIfPangram(String sentence) {

}
}
```

Scala Solution:

```
object Solution {
def checkIfPangram(sentence: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec check_if_pangram(sentence :: String.t) :: boolean
def check_if_pangram(sentence) do

end
end
```

Erlang Solution:

```
-spec check_if_pangram(Sentence :: unicode:unicode_binary()) -> boolean().
check_if_pangram(Sentence) ->
.
```

Racket Solution:

```
(define/contract (check-if-pangram sentence)
(-> string? boolean?)
)
```