Problem 2490: Circular Sentence

Problem Information

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

Problem Description

A

sentence

is a list of words that are separated by a

single

space with no leading or trailing spaces.

For example,

"Hello World"

,

"HELLO"

,

"hello world hello world"

are all sentences.

Words consist of

only

uppercase and lowercase English letters. Uppercase and lowercase English letters are considered different.

A sentence is

circular

if:

The last character of each word in the sentence is equal to the first character of its next word.

The last character of the last word is equal to the first character of the first word.

For example,

"leetcode exercises sound delightful"

,

"eetcode"

,

"leetcode eats soul"

are all circular sentences. However,

"Leetcode is cool"

,

"happy Leetcode"

,

"Leetcode"

and

"I like Leetcode"

are

not

circular sentences.

Given a string

sentence

, return

true

if it is circular

. Otherwise, return

false

.

Example 1:

Input:

sentence = "leetcode exercises sound delightful"

Output:

true

Explanation:

The words in sentence are ["leetcode", "exercises", "sound", "delightful"]. - leetcod e

's last character is equal to

e

xercises's first character. - exercise

s

's last character is equal to

s

ound's first character. - soun

d

's last character is equal to

d

elightful's first character. - delightfu

l

's last character is equal to

l

eetcode's first character. The sentence is circular.

Example 2:

Input:

sentence = "eetcode"

Output:

true

Explanation:

The words in sentence are ["eetcode"]. - eetcod

e

's last character is equal to

e

etcode's first character. The sentence is circular.

Example 3:

Input:

sentence = "Leetcode is cool"

Output:

false

Explanation:

The words in sentence are ["Leetcode", "is", "cool"]. - Leetcod

e

's last character is

not

equal to

i

s's first character. The sentence is

not

circular.

Constraints:

1 <= sentence.length <= 500

sentence

consist of only lowercase and uppercase English letters and spaces.

The words in

sentence

are separated by a single space.

There are no leading or trailing spaces.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
bool isCircularSentence(char* sentence) {

}
```

Go:

```
func isCircularSentence(sentence string) bool {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (is-circular-sentence sentence)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Circular Sentence
* 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 isCircularSentence(string sentence) {

}
};
```

Java Solution:

```
/**
* Problem: Circular Sentence
* 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 isCircularSentence(String sentence) {
}
}
```

Python3 Solution:

```
"""
Problem: Circular Sentence
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 isCircularSentence(self, sentence: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Circular Sentence
* 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} sentence
* @return {boolean}
*/
var isCircularSentence = function(sentence) {

};
```

TypeScript Solution:

```
/**
* Problem: Circular Sentence
* 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 isCircularSentence(sentence: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Circular Sentence
* 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 IsCircularSentence(string sentence) {

}
}
```

C Solution:

```
/*
* Problem: Circular Sentence
* 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 isCircularSentence(char* sentence) {

}
```

Go Solution:

```
// Problem: Circular Sentence
// 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 isCircularSentence(sentence string) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Circular Sentence
// 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_circular_sentence(sentence: String) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (is-circular-sentence sentence)
(-> string? boolean?)
)
```