Problem 796: Rotate String

Problem Information

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

Problem Description

Given two strings

s

and

goal

, return

true

if and only if

s

can become

goal

after some number of

shifts

on

s

.

A

shift

on

s

consists of moving the leftmost character of

s

to the rightmost position.

For example, if

s = "abcde"

, then it will be

"bcdea"

after one shift.

Example 1:

Input:

s = "abcde", goal = "cdeab"

Output:

true

Example 2:

Input:

s = "abcde", goal = "abced"

Output:

false

Constraints:

1 <= s.length, goal.length <= 100

s

and

goal

consist of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
bool rotateString(string s, string goal) {

}
};
```

Java:

```
class Solution {
public boolean rotateString(String s, String goal) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function rotateString(s: string, goal: string): boolean {

};
```

C#:

```
public class Solution {
public bool RotateString(string s, string goal) {

}
}
```

C:

```
bool rotateString(char* s, char* goal) {

}
```

Go:

```
func rotateString(s string, goal string) bool {

}
```

Kotlin:

```
class Solution {
fun rotateString(s: String, goal: String): Boolean {

}
}
```

Swift:

```
class Solution {
func rotateString(_ s: String, _ goal: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn rotate_string(s: String, goal: String) -> bool {

}
}
```

Ruby:

```
# @param {String} s
# @param {String} goal
# @return {Boolean}
def rotate_string(s, goal)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param String $goal
* @return Boolean
*/
function rotateString($s, $goal) {

}
}
```

Dart:

```
class Solution {
bool rotateString(String s, String goal) {

}
}
```

Scala:

```
object Solution {
def rotateString(s: String, goal: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec rotate_string(s :: String.t, goal :: String.t) :: boolean
def rotate_string(s, goal) do

end
end
```

Erlang:

```
-spec rotate_string(S :: unicode:unicode_binary(), Goal ::
unicode:unicode_binary()) -> boolean().
```

```
rotate_string(S, Goal) ->
.
```

Racket:

```
(define/contract (rotate-string s goal)
(-> string? string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Rotate 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 rotateString(string s, string goal) {

}
};
```

Java Solution:

```
/**
* Problem: Rotate 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 rotateString(String s, String goal) {

}
}
```

Python3 Solution:

```
"""
Problem: Rotate 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 rotateString(self, s: str, goal: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Rotate 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} s
* @param {string} goal
* @return {boolean}
*/
var rotateString = function(s, goal) {

};
```

TypeScript Solution:

```
/**
* Problem: Rotate 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 rotateString(s: string, goal: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Rotate 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 RotateString(string s, string goal) {

}
}
```

C Solution:

```
/*
* Problem: Rotate 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 rotateString(char* s, char* goal) {

}
```

Go Solution:

```
// Problem: Rotate 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 rotateString(s string, goal string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun rotateString(s: String, goal: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func rotateString(_ s: String, _ goal: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Rotate 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 rotate_string(s: String, goal: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {String} goal
# @return {Boolean}
def rotate_string(s, goal)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param String $goal
* @return Boolean
*/
function rotateString($s, $goal) {

}
}
```

Dart Solution:

```
class Solution {
bool rotateString(String s, String goal) {

}
}
```

Scala Solution:

```
object Solution {
def rotateString(s: String, goal: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec rotate_string(s :: String.t, goal :: String.t) :: boolean
def rotate_string(s, goal) do

end
end
```

Erlang Solution:

```
-spec rotate_string(S :: unicode:unicode_binary(), Goal ::
unicode:unicode_binary()) -> boolean().
rotate_string(S, Goal) ->
```

```
.
```

Racket Solution:

```
(define/contract (rotate-string s goal)
(-> string? string? boolean?)
)
```