Problem 3083: Existence of a Substring in a String and Its Reverse

Problem Information

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

Problem Description

Given a

string

s

, find any

substring

of length

2

which is also present in the reverse of

s

.

Return

true

if such a substring exists, and

false

otherwise.

Example 1:

Input:

s = "leetcode"

Output:

true

Explanation:

Substring

"ee"

is of length

2

which is also present in

reverse(s) == "edocteel"

.

Example 2:

Input:

s = "abcba"

Output:

true

Explanation:

All of the substrings of length

2

"ab"

,

"bc"

,

"cb"

,

"ba"

are also present in

reverse(s) == "abcba"

.

Example 3:

Input:

s = "abcd"

Output:

false

Explanation:

There is no substring of length

2

in

s

, which is also present in the reverse of

s

.

Constraints:

1 <= s.length <= 100

s

consists only of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
```

```
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
bool isSubstringPresent(char* s) {

}
```

Go:

```
func isSubstringPresent(s string) bool {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (is-substring-present s)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Existence of a Substring in a String and Its Reverse
* Difficulty: Easy
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
bool isSubstringPresent(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Existence of a Substring in a String and Its Reverse
* Difficulty: Easy
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public boolean isSubstringPresent(String s) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Existence of a Substring in a String and Its Reverse
Difficulty: Easy
Tags: string, tree, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Existence of a Substring in a String and Its Reverse
* Difficulty: Easy
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Existence of a Substring in a String and Its Reverse
* Difficulty: Easy
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function isSubstringPresent(s: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Existence of a Substring in a String and Its Reverse
* Difficulty: Easy
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public bool IsSubstringPresent(string s) {

}
}
```

C Solution:

```
/*
* Problem: Existence of a Substring in a String and Its Reverse
* Difficulty: Easy
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

bool isSubstringPresent(char* s) {

}
```

Go Solution:

```
// Problem: Existence of a Substring in a String and Its Reverse
// Difficulty: Easy
// Tags: string, tree, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func isSubstringPresent(s string) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Existence of a Substring in a String and Its Reverse
// Difficulty: Easy
// Tags: string, tree, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (is-substring-present s)
(-> string? boolean?)
)
```