Problem 1790: Check if One String Swap Can Make Strings Equal

Problem Information

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

Problem Description

You are given two strings

s1

and

s2

of equal length. A

string swap

is an operation where you choose two indices in a string (not necessarily different) and swap the characters at these indices.

Return

true

if it is possible to make both strings equal by performing

at most one string swap

on

exactly one

of the strings.

Otherwise, return

false

.

Example 1:

Input:

s1 = "bank", s2 = "kanb"

Output:

true

Explanation:

For example, swap the first character with the last character of s2 to make "bank".

Example 2:

Input:

s1 = "attack", s2 = "defend"

Output:

false

Explanation:

It is impossible to make them equal with one string swap.

Example 3:

Input:

s1 = "kelb", s2 = "kelb"

Output:

true

Explanation:

The two strings are already equal, so no string swap operation is required.

Constraints:

1 <= s1.length, s2.length <= 100

s1.length == s2.length

s1

and

s2

consist of only lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
bool areAlmostEqual(string s1, string s2) {

}
};
```

Java:

```
class Solution {
public boolean areAlmostEqual(String s1, String s2) {

}
}
```

Python3:

```
class Solution:
def areAlmostEqual(self, s1: str, s2: str) -> bool:
```

Python:

```
class Solution(object):
def areAlmostEqual(self, s1, s2):
"""
:type s1: str
:type s2: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} s1
* @param {string} s2
* @return {boolean}
*/
var areAlmostEqual = function(s1, s2) {

};
```

TypeScript:

```
function areAlmostEqual(s1: string, s2: string): boolean {

};
```

C#:

```
public class Solution {
public bool AreAlmostEqual(string s1, string s2) {
```

```
}
}
```

C:

```
bool areAlmostEqual(char* s1, char* s2) {

}
```

Go:

```
func areAlmostEqual(s1 string, s2 string) bool {

}
```

Kotlin:

```
class Solution {
fun areAlmostEqual(s1: String, s2: String): Boolean {

}
}
```

Swift:

```
class Solution {
func areAlmostEqual(_ s1: String, _ s2: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn are_almost_equal(s1: String, s2: String) -> bool {

}
}
```

Ruby:

```
# @param {String} s1
# @param {String} s2
# @return {Boolean}
def are_almost_equal(s1, s2)

end
```

PHP:

```
class Solution {

/**
* @param String $s1
* @param String $s2
* @return Boolean
*/
function areAlmostEqual($s1, $s2) {

}
}
```

Dart:

```
class Solution {
bool areAlmostEqual(String s1, String s2) {

}
}
```

Scala:

```
object Solution {
def areAlmostEqual(s1: String, s2: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec are_almost_equal(s1 :: String.t, s2 :: String.t) :: boolean
def are_almost_equal(s1, s2) do
```

```
end
end
```

Erlang:

```
-spec are_almost_equal(S1 :: unicode:unicode_binary(), S2 ::
unicode:unicode_binary()) -> boolean().
are_almost_equal(S1, S2) ->
.
```

Racket:

```
(define/contract (are-almost-equal s1 s2)
(-> string? string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if One String Swap Can Make Strings Equal
* 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 areAlmostEqual(string s1, string s2) {

}
};
```

Java Solution:

```
/**
* Problem: Check if One String Swap Can Make Strings Equal
* 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 areAlmostEqual(String s1, String s2) {

}
}
```

Python3 Solution:

```
"""
Problem: Check if One String Swap Can Make Strings Equal
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 areAlmostEqual(self, s1: str, s2: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def areAlmostEqual(self, s1, s2):
"""
:type s1: str
:type s2: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Check if One String Swap Can Make Strings Equal
* 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} s1
* @param {string} s2
* @return {boolean}
*/
var areAlmostEqual = function(s1, s2) {

};
```

TypeScript Solution:

```
/**
* Problem: Check if One String Swap Can Make Strings Equal
* 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 areAlmostEqual(s1: string, s2: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if One String Swap Can Make Strings Equal
* 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 AreAlmostEqual(string s1, string s2) {

}
}
```

C Solution:

```
/*
* Problem: Check if One String Swap Can Make Strings Equal
* 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 areAlmostEqual(char* s1, char* s2) {

}
```

Go Solution:

```
// Problem: Check if One String Swap Can Make Strings Equal
// 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 areAlmostEqual(s1 string, s2 string) bool {
```

```
}
```

Kotlin Solution:

```
class Solution {
fun areAlmostEqual(s1: String, s2: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func areAlmostEqual(_ s1: String, _ s2: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Check if One String Swap Can Make Strings Equal
// 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_almost_equal(s1: String, s2: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} s1
# @param {String} s2
# @return {Boolean}
def are_almost_equal(s1, s2)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param String $s1
* @param String $s2
* @return Boolean
*/
function areAlmostEqual($s1, $s2) {

}
}
```

Dart Solution:

```
class Solution {
bool areAlmostEqual(String s1, String s2) {

}
}
```

Scala Solution:

```
object Solution {
def areAlmostEqual(s1: String, s2: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec are_almost_equal(s1 :: String.t, s2 :: String.t) :: boolean
def are_almost_equal(s1, s2) do

end
end
```

Erlang Solution:

```
-spec are_almost_equal(S1 :: unicode:unicode_binary(), S2 ::
unicode:unicode_binary()) -> boolean().
are_almost_equal(S1, S2) ->
.
```

Racket Solution:

```
(define/contract (are-almost-equal s1 s2)
(-> string? string? boolean?)
)
```