Problem 2839: Check if Strings Can be Made Equal With Operations I

Problem Information

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

Problem Description

You are given two strings

s1

and

s2

, both of length

4

, consisting of

lowercase

English letters.

You can apply the following operation on any of the two strings

any

number of times:

Choose any two indices

i

and

j

such that

j - i = 2

, then

swap

the two characters at those indices in the string.

Return

true

if you can make the strings

s1

and

s2

equal, and

false

otherwise

.

Example 1:

Input:

s1 = "abcd", s2 = "cdab"

Output:

true

Explanation:

We can do the following operations on s1: - Choose the indices i = 0, j = 2. The resulting string is s1 = "cbad". - Choose the indices i = 1, j = 3. The resulting string is s1 = "cdab" = s2.

Example 2:

Input:

s1 = "abcd", s2 = "dacb"

Output:

false

Explanation:

It is not possible to make the two strings equal.

Constraints:

s1.length == s2.length == 4

s1

and

s2

consist only of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (can-be-equal s1 s2)
(-> string? string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if Strings Can be Made Equal With Operations I
* 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 canBeEqual(string s1, string s2) {

}
};
```

Java Solution:

```
/**
* Problem: Check if Strings Can be Made Equal With Operations I
* 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 canBeEqual(String s1, String s2) {

}
}
```

Python3 Solution:

```
"""
Problem: Check if Strings Can be Made Equal With Operations I
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 canBeEqual(self, s1: str, s2: str) -> bool:
```

```
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check if Strings Can be Made Equal With Operations I
* 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} s1
* @param {string} s2
* @return {boolean}
*/
var canBeEqual = function(s1, s2) {

};
```

TypeScript Solution:

```
/**
* Problem: Check if Strings Can be Made Equal With Operations I
* 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 canBeEqual(s1: string, s2: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if Strings Can be Made Equal With Operations I
* 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 CanBeEqual(string s1, string s2) {

}
}
```

C Solution:

```
/*
* Problem: Check if Strings Can be Made Equal With Operations I
* 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 canBeEqual(char* s1, char* s2) {

}
```

Go Solution:

```go
// Problem: Check if Strings Can be Made Equal With Operations I
// 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 canBeEqual(s1 string, s2 string) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```rust
// Problem: Check if Strings Can be Made Equal With Operations I
// 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 can_be_equal(s1: String, s2: String) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (can-be-equal s1 s2)
(-> string? string? boolean?)
)
```