Problem 2840: Check if Strings Can be Made Equal With Operations II

Problem Information

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

Problem Description

You are given two strings

s1

and

s2

, both of length

n

, 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

i < j

and the difference

j - i

is

even

, 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 = "abcdba", s2 = "cabdab"

Output:

true

Explanation:

We can apply the following operations on s1: - Choose the indices i = 0, j = 2. The resulting string is s1 = "cbadba". - Choose the indices i = 2, j = 4. The resulting string is s1 = "cbbdaa". - Choose the indices i = 1, j = 5. The resulting string is s1 = "cabdab" = s2.

Example 2:

Input:

s1 = "abe", s2 = "bea"

Output:

false

Explanation:

It is not possible to make the two strings equal.

Constraints:

n == s1.length == s2.length

1 <= n <= 10

5

s1

and

s2

consist only of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param String $s1
```

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (check-strings s1 s2)
(-> string? string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if Strings Can be Made Equal With Operations II
* Difficulty: Medium
* Tags: string, hash, sort
*
* 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 checkStrings(string s1, string s2) {

}
};
```

Java Solution:

```
/**
* Problem: Check if Strings Can be Made Equal With Operations II
* Difficulty: Medium
* Tags: string, hash, sort
*
* 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 checkStrings(String s1, String s2) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Check if Strings Can be Made Equal With Operations II
Difficulty: Medium
Tags: string, hash, sort

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 checkStrings(self, s1: str, s2: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check if Strings Can be Made Equal With Operations II
* Difficulty: Medium
* Tags: string, hash, sort
*
* 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 checkStrings = function(s1, s2) {

};
```

TypeScript Solution:

```
/**
* Problem: Check if Strings Can be Made Equal With Operations II
* Difficulty: Medium
* Tags: string, hash, sort
*
* 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 checkStrings(s1: string, s2: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if Strings Can be Made Equal With Operations II
* Difficulty: Medium
* Tags: string, hash, sort
*
* 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 CheckStrings(string s1, string s2) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Check if Strings Can be Made Equal With Operations II
* Difficulty: Medium
* Tags: string, hash, sort
*
* 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 checkStrings(char* s1, char* s2) {

}
```

Go Solution:

```
// Problem: Check if Strings Can be Made Equal With Operations II
// Difficulty: Medium
// Tags: string, hash, sort
//
// 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 checkStrings(s1 string, s2 string) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Check if Strings Can be Made Equal With Operations II
// Difficulty: Medium
// Tags: string, hash, sort
//
// 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 check_strings(s1: String, s2: String) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

```
}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (check-strings s1 s2)
(-> string? string? boolean?)
)
```