Problem 1247: Minimum Swaps to Make Strings Equal

Problem Information

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

Problem Description

You are given two strings

s1

and

s2

of equal length consisting of letters

"x"

and

"y"

only

. Your task is to make these two strings equal to each other. You can swap any two characters that belong to

different

strings, which means: swap

s1[i]

and

s2[j]

.

Return the minimum number of swaps required to make

s1

and

s2

equal, or return

-1

if it is impossible to do so.

Example 1:

Input:

s1 = "xx", s2 = "yy"

Output:

1

Explanation:

Swap s1[0] and s2[1], s1 = "yx", s2 = "yx".

Example 2:

Input:

s1 = "xy", s2 = "yx"

Output:

2

Explanation:

Swap s1[0] and s2[0], s1 = "yy", s2 = "xx". Swap s1[0] and s2[1], s1 = "xy", s2 = "xy". Note that you cannot swap s1[0] and s1[1] to make s1 equal to "yx", cause we can only swap chars in different strings.

Example 3:

Input:

s1 = "xx", s2 = "xy"

Output:

-1

Constraints:

1 <= s1.length, s2.length <= 1000

s1.length == s2.length

s1, s2

only contain

'x'

or

'y'

.

Code Snippets

C++:

```
class Solution {
public:
int minimumSwap(string s1, string s2) {

}
};
```

Java:

```
class Solution {
public int minimumSwap(String s1, String s2) {

}
}
```

Python3:

```
class Solution:
def minimumSwap(self, s1: str, s2: str) -> int:
```

Python:

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

JavaScript:

```
/**
* @param {string} s1
* @param {string} s2
* @return {number}
*/
```

```
var minimumSwap = function(s1, s2) {

};
```

TypeScript:

```
function minimumSwap(s1: string, s2: string): number {

};
```

C#:

```
public class Solution {
public int MinimumSwap(string s1, string s2) {

}
}
```

C:

```
int minimumSwap(char* s1, char* s2) {

}
```

Go:

```
func minimumSwap(s1 string, s2 string) int {

}
```

Kotlin:

```
class Solution {
fun minimumSwap(s1: String, s2: String): Int {

}
}
```

Swift:

```
class Solution {
func minimumSwap(_ s1: String, _ s2: String) -> Int {



}
}
```

Rust:

```
impl Solution {
pub fn minimum_swap(s1: String, s2: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s1
# @param {String} s2
# @return {Integer}
def minimum_swap(s1, s2)

end
```

PHP:

```
class Solution {

/**
* @param String $s1
* @param String $s2
* @return Integer
*/
function minimumSwap($s1, $s2) {

}
}
```

Dart:

```
class Solution {
int minimumSwap(String s1, String s2) {

}
}
```

Scala:

```
object Solution {
def minimumSwap(s1: String, s2: String): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec minimum_swap(S1 :: unicode:unicode_binary(), S2 ::
unicode:unicode_binary()) -> integer().
minimum_swap(S1, S2) ->
.
```

Racket:

```
(define/contract (minimum-swap s1 s2)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Swaps to Make Strings Equal
* Difficulty: Medium
* Tags: string, greedy, math
*
* 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:
int minimumSwap(string s1, string s2) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Swaps to Make Strings Equal
* Difficulty: Medium
* Tags: string, greedy, math
*
* 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 int minimumSwap(String s1, String s2) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Swaps to Make Strings Equal
Difficulty: Medium
Tags: string, greedy, math

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Swaps to Make Strings Equal
* Difficulty: Medium
* Tags: string, greedy, math
*
* 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 {number}
*/
var minimumSwap = function(s1, s2) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Swaps to Make Strings Equal
* Difficulty: Medium
* Tags: string, greedy, math
```

```
*
* 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 minimumSwap(s1: string, s2: string): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Swaps to Make Strings Equal
* Difficulty: Medium
* Tags: string, greedy, math
*
* 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 int MinimumSwap(string s1, string s2) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Swaps to Make Strings Equal
* Difficulty: Medium
* Tags: string, greedy, math
*
* 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
*/

int minimumSwap(char* s1, char* s2) {
}
```

Go Solution:

```go
// Problem: Minimum Swaps to Make Strings Equal
// Difficulty: Medium
// Tags: string, greedy, math
//
// 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 minimumSwap(s1 string, s2 string) int {

}
```

Kotlin Solution:

```kotlin
class Solution {
fun minimumSwap(s1: String, s2: String): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```rust
// Problem: Minimum Swaps to Make Strings Equal
// Difficulty: Medium
// Tags: string, greedy, math
//
// 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 minimum_swap(s1: String, s2: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s1
# @param {String} s2
# @return {Integer}
def minimum_swap(s1, s2)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s1
* @param String $s2
* @return Integer
*/
function minimumSwap($s1, $s2) {

}
}
```

Dart Solution:

```
class Solution {
int minimumSwap(String s1, String s2) {

}
}
```

Scala Solution:

```
object Solution {
def minimumSwap(s1: String, s2: String): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec minimum_swap(S1 :: unicode:unicode_binary(), S2 ::
unicode:unicode_binary()) -> integer().
minimum_swap(S1, S2) ->
.
```

Racket Solution:

```
(define/contract (minimum-swap s1 s2)
(-> string? string? exact-integer?)
)
```