Problem 87: Scramble String

Problem Information

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

Problem Description

We can scramble a string s to get a string t using the following algorithm:

If the length of the string is 1, stop.

If the length of the string is > 1, do the following:

Split the string into two non-empty substrings at a random index, i.e., if the string is

s

, divide it to

x

and

y

where

s = x + y

.

Randomly

decide to swap the two substrings or to keep them in the same order. i.e., after this step,

s

may become

s = x + y

or

s = y + x

.

Apply step 1 recursively on each of the two substrings

x

and

y

.

Given two strings

s1

and

s2

of

the same length

, return

true

if

s2

is a scrambled string of

s1

, otherwise, return

false

.

Example 1:

Input:

s1 = "great", s2 = "rgeat"

Output:

true

Explanation:

One possible scenario applied on s1 is: "great" --> "gr/eat" // divide at random index. "gr/eat" --> "gr/eat" // random decision is not to swap the two substrings and keep them in order. "gr/eat" --> "g/r / e/at" // apply the same algorithm recursively on both substrings. divide at random index each of them. "g/r / e/at" --> "r/g / e/at" // random decision was to swap the first substring and to keep the second substring in the same order. "r/g / e/at" --> "r/g / e/ a/t" // again apply the algorithm recursively, divide "at" to "a/t". "r/g / e/ a/t" --> "r/g / e/ a/t" // random decision is to keep both substrings in the same order. The algorithm stops now, and the result string is "rgeat" which is s2. As one possible scenario led s1 to be scrambled to s2, we return true.

Example 2:

Input:

s1 = "abcde", s2 = "caebd"

Output:

false

Example 3:

Input:

s1 = "a", s2 = "a"

Output:

true

Constraints:

s1.length == s2.length

1 <= s1.length <= 30

s1

and

s2

consist of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
bool isScramble(string s1, string s2) {
}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
def is_scramble(s1, s2) do

end
end
```

Erlang:

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

Racket:

```
(define/contract (is-scramble s1 s2)
(-> string? string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Scramble String
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Scramble String
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public boolean isScramble(String s1, String s2) {

}
}
```

Python3 Solution:

```
"""
Problem: Scramble String
Difficulty: Hard
Tags: string, tree, dp

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def isScramble(self, s1: str, s2: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Scramble String
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Scramble String
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function isScramble(s1: string, s2: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Scramble String
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public bool IsScramble(string s1, string s2) {

}
}
```

C Solution:

```
/*
* Problem: Scramble String
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

bool isScramble(char* s1, char* s2) {

}
```

Go Solution:

```
// Problem: Scramble String
// Difficulty: Hard
// Tags: string, tree, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func isScramble(s1 string, s2 string) bool {
}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Scramble String
// Difficulty: Hard
// Tags: string, tree, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (is-scramble s1 s2)
(-> string? string? boolean?)
)
```