Problem 97: Interleaving String

Problem Information

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

Problem Description

Given strings

s1

,

s2

, and

s3

, find whether

s3

is formed by an

interleaving

of

s1

and

s2

.

An

interleaving

of two strings

s

and

t

is a configuration where

s

and

t

are divided into

n

and

m

substrings

respectively, such that:

s = s

1

+ s

2

+ ... + s

n

t = t

1

+ t

2

+ ... + t

m

|n - m| <= 1

The

interleaving

is

s

1

+ t

1

+ s

2

+ t

2

+ s

3

+ t

3

+ ...

or

t

1

+ s

1

+ t

2

+ s

2

+ t

3

+ s

3

+ ...

Note:

a + b

is the concatenation of strings

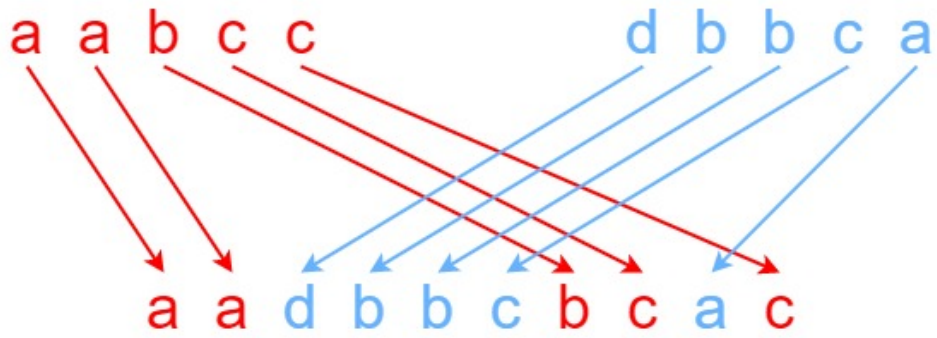
a

and

b

.

Example 1:



Input:

s1 = "aabcc", s2 = "dbbca", s3 = "aadbbcbcac"

Output:

true

Explanation:

One way to obtain s3 is: Split s1 into s1 = "aa" + "bc" + "c", and s2 into s2 = "dbbc" + "a". Interleaving the two splits, we get "aa" + "dbbc" + "bc" + "a" + "c" = "aadbbcbcac". Since s3 can be obtained by interleaving s1 and s2, we return true.

Example 2:

Input:

s1 = "aabcc", s2 = "dbbca", s3 = "aadbbbaccc"

Output:

false

Explanation:

Notice how it is impossible to interleave s2 with any other string to obtain s3.

Example 3:

Input:

s1 = "", s2 = "", s3 = ""

Output:

true

Constraints:

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

0 <= s3.length <= 200

s1

,

s2

, and

s3

consist of lowercase English letters.

Follow up:

Could you solve it using only

O(s2.length)

additional memory space?

Code Snippets

C++:

```
class Solution {
public:
bool isInterleave(string s1, string s2, string s3) {

}
};
```

Java:

```
class Solution {
public boolean isInterleave(String s1, String s2, String s3) {

}
}
```

Python3:

```
class Solution:
def isInterleave(self, s1: str, s2: str, s3: str) -> bool:
```

Python:

```
class Solution(object):
def isInterleave(self, s1, s2, s3):
"""
:type s1: str
:type s2: str
:type s3: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} s1
* @param {string} s2
* @param {string} s3
* @return {boolean}
*/
var isInterleave = function(s1, s2, s3) {

};
```

TypeScript:

```
function isInterleave(s1: string, s2: string, s3: string): boolean {

};
```

C#:

```
public class Solution {
public bool IsInterleave(string s1, string s2, string s3) {

}
}
```

C:

```
bool isInterleave(char* s1, char* s2, char* s3) {

}
```

Go:

```
func isInterleave(s1 string, s2 string, s3 string) bool {

}
```

Kotlin:

```
class Solution {
fun isInterleave(s1: String, s2: String, s3: String): Boolean {

}
}
```

Swift:

```
class Solution {
func isInterleave(_ s1: String, _ s2: String, _ s3: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_interleave(s1: String, s2: String, s3: String) -> bool {

}
}
```

Ruby:

```
# @param {String} s1
# @param {String} s2
# @param {String} s3
# @return {Boolean}
def is_interleave(s1, s2, s3)

end
```

PHP:

```
class Solution {

/**
```

```
* @param String $s1
* @param String $s2
* @param String $s3
* @return Boolean
*/
function isInterleave($s1, $s2, $s3) {

}
}
```

Dart:

```
class Solution {
bool isInterleave(String s1, String s2, String s3) {

}
}
```

Scala:

```
object Solution {
def isInterleave(s1: String, s2: String, s3: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_interleave(s1 :: String.t, s2 :: String.t, s3 :: String.t) ::
boolean
def is_interleave(s1, s2, s3) do

end
end
```

Erlang:

```
-spec is_interleave(S1 :: unicode:unicode_binary(), S2 ::
unicode:unicode_binary(), S3 :: unicode:unicode_binary()) -> boolean().
is_interleave(S1, S2, S3) ->
.
```

Racket:

```
(define/contract (is-interleave s1 s2 s3)
(-> string? string? string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Interleaving String
* Difficulty: Medium
* 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 isInterleave(string s1, string s2, string s3) {

}
};
```

Java Solution:

```
/**
* Problem: Interleaving String
* Difficulty: Medium
* 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 isInterleave(String s1, String s2, String s3) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Interleaving String
Difficulty: Medium
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 isInterleave(self, s1: str, s2: str, s3: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def isInterleave(self, s1, s2, s3):
"""
:type s1: str
:type s2: str
:type s3: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Interleaving String
* Difficulty: Medium
* 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
* @param {string} s3
* @return {boolean}
*/
var isInterleave = function(s1, s2, s3) {

};
```

TypeScript Solution:

```
/**
* Problem: Interleaving String
* Difficulty: Medium
* 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 isInterleave(s1: string, s2: string, s3: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Interleaving String
* Difficulty: Medium
* 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 IsInterleave(string s1, string s2, string s3) {

}
}
```

C Solution:

```
/*
* Problem: Interleaving String
* Difficulty: Medium
* 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 isInterleave(char* s1, char* s2, char* s3) {

}
```

Go Solution:

```
// Problem: Interleaving String
// Difficulty: Medium
// 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 isInterleave(s1 string, s2 string, s3 string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isInterleave(s1: String, s2: String, s3: String): Boolean {
```

```
}
}
```

Swift Solution:

```
class Solution {
func isInterleave(_ s1: String, _ s2: String, _ s3: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Interleaving String
// Difficulty: Medium
// 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_interleave(s1: String, s2: String, s3: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} s1
# @param {String} s2
# @param {String} s3
# @return {Boolean}
def is_interleave(s1, s2, s3)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String $s1
* @param String $s2
* @param String $s3
* @return Boolean
*/
function isInterleave($s1, $s2, $s3) {

}
}
```

Dart Solution:

```
class Solution {
bool isInterleave(String s1, String s2, String s3) {

}
}
```

Scala Solution:

```
object Solution {
def isInterleave(s1: String, s2: String, s3: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_interleave(s1 :: String.t, s2 :: String.t, s3 :: String.t) ::
boolean
def is_interleave(s1, s2, s3) do

end
end
```

Erlang Solution:

```
-spec is_interleave(S1 :: unicode:unicode_binary(), S2 ::
unicode:unicode_binary(), S3 :: unicode:unicode_binary()) -> boolean().
```

```
is_interleave(S1, S2, S3) ->
.
```

Racket Solution:

```
(define/contract (is-interleave s1 s2 s3)
(-> string? string? string? boolean?)
)
```