Problem 161: One Edit Distance

Problem Information

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

Problem Description

Given two strings

s

and

t

, return

true

if they are both one edit distance apart, otherwise return

false

.

A string

s

is said to be one distance apart from a string

t

if you can:

Insert

exactly one

character into

s

to get

t

.

Delete

exactly one

character from

s

to get

t

.

Replace

exactly one

character of

s

with

a different character

to get

t

.

Example 1:

Input:

s = "ab", t = "acb"

Output:

true

Explanation:

We can insert 'c' into s to get t.

Example 2:

Input:

s = "", t = ""

Output:

false

Explanation:

We cannot get t from s by only one step.

Constraints:

0 <= s.length, t.length <= 10

4

s

and

t

consist of lowercase letters, uppercase letters, and digits.

Code Snippets

C++:

```
class Solution {
public:
bool isOneEditDistance(string s, string t) {

}
};
```

Java:

```
class Solution {
public boolean isOneEditDistance(String s, String t) {

}
}
```

Python3:

```
class Solution:
def isOneEditDistance(self, s: str, t: str) -> bool:
```

Python:

```
class Solution(object):
def isOneEditDistance(self, s, t):
```

```
"""
:type s: str
:type t: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {string} t
* @return {boolean}
*/
var isOneEditDistance = function(s, t) {

};
```

TypeScript:

```
function isOneEditDistance(s: string, t: string): boolean {

};
```

C#:

```
public class Solution {
public bool IsOneEditDistance(string s, string t) {

}
}
```

C:

```
bool isOneEditDistance(char* s, char* t) {

}
```

Go:

```
func isOneEditDistance(s string, t string) bool {

}
```

Kotlin:

```
class Solution {
fun isOneEditDistance(s: String, t: String): Boolean {

}
}
```

Swift:

```
class Solution {
func isOneEditDistance(_ s: String, _ t: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_one_edit_distance(s: String, t: String) -> bool {

}
}
```

Ruby:

```
# @param {String} s
# @param {String} t
# @return {Boolean}
def is_one_edit_distance(s, t)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param String $t
* @return Boolean
*/
function isOneEditDistance($s, $t) {
```

```
}
}
```

Dart:

```
class Solution {
bool isOneEditDistance(String s, String t) {

}
}
```

Scala:

```
object Solution {
def isOneEditDistance(s: String, t: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_one_edit_distance(s :: String.t, t :: String.t) :: boolean
def is_one_edit_distance(s, t) do

end
end
```

Erlang:

```
-spec is_one_edit_distance(S :: unicode:unicode_binary(), T ::
unicode:unicode_binary()) -> boolean().
is_one_edit_distance(S, T) ->
.
```

Racket:

```
(define/contract (is-one-edit-distance s t)
(-> string? string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: One Edit Distance
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
bool isOneEditDistance(string s, string t) {

}
};
```

Java Solution:

```
/**
* Problem: One Edit Distance
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public boolean isOneEditDistance(String s, String t) {

}
}
```

Python3 Solution:

```
"""
Problem: One Edit Distance
Difficulty: Medium
Tags: array, string

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def isOneEditDistance(self, s: str, t: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def isOneEditDistance(self, s, t):
"""
:type s: str
:type t: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: One Edit Distance
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} s
* @param {string} t
* @return {boolean}
*/
```

```
var isOneEditDistance = function(s, t) {

};
```

TypeScript Solution:

```
/**
* Problem: One Edit Distance
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function isOneEditDistance(s: string, t: string): boolean {

};
```

C# Solution:

```
/*
* Problem: One Edit Distance
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public bool IsOneEditDistance(string s, string t) {

}
}
```

C Solution:

```
/*
* Problem: One Edit Distance
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

bool isOneEditDistance(char* s, char* t) {

}
```

Go Solution:

```
// Problem: One Edit Distance
// Difficulty: Medium
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func isOneEditDistance(s string, t string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isOneEditDistance(s: String, t: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isOneEditDistance(_ s: String, _ t: String) -> Bool {

}
```

```
}
```

Rust Solution:

```
// Problem: One Edit Distance
// Difficulty: Medium
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn is_one_edit_distance(s: String, t: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {String} t
# @return {Boolean}
def is_one_edit_distance(s, t)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param String $t
* @return Boolean
*/
function isOneEditDistance($s, $t) {

}
}
```

Dart Solution:

```
class Solution {
bool isOneEditDistance(String s, String t) {

}
}
```

Scala Solution:

```
object Solution {
def isOneEditDistance(s: String, t: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_one_edit_distance(s :: String.t, t :: String.t) :: boolean
def is_one_edit_distance(s, t) do

end
end
```

Erlang Solution:

```
-spec is_one_edit_distance(S :: unicode:unicode_binary(), T ::
unicode:unicode_binary()) -> boolean().
is_one_edit_distance(S, T) ->
.
```

Racket Solution:

```
(define/contract (is-one-edit-distance s t)
(-> string? string? boolean?)
)
```