Problem 1638: Count Substrings That Differ by One Character

Problem Information

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

Problem Description

Given two strings

s

and

t

, find the number of ways you can choose a non-empty substring of

s

and replace a

single character

by a different character such that the resulting substring is a substring of

t

. In other words, find the number of substrings in

s

that differ from some substring in

t

by

exactly

one character.

For example, the underlined substrings in

"

compute

r"

and

"

computa

tion"

only differ by the

'e'

/

'a'

, so this is a valid way.

Return

the number of substrings that satisfy the condition above.

A

substring

is a contiguous sequence of characters within a string.

Example 1:

Input:

s = "aba", t = "baba"

Output:

6

Explanation:

The following are the pairs of substrings from s and t that differ by exactly 1 character: ("

a

ba", "

b

aba") ("

a

ba", "ba

b

a") ("ab

a

", "

b

aba") ("ab

a

", "ba

b

a") ("a

b

a", "b

a

ba") ("a

b

a", "bab

a

") The underlined portions are the substrings that are chosen from s and t.

Example 2:

Input:

s = "ab", t = "bb"

Output:

3

Explanation:

The following are the pairs of substrings from s and t that differ by 1 character: ("

a

b", "

b

b") ("

a

b", "b

b

") ("

ab

", "

bb

") The underlined portions are the substrings that are chosen from s and t.

Constraints:

1 <= s.length, t.length <= 100

s

and

t

consist of lowercase English letters only.

Code Snippets

C++:

```
class Solution {
public:
int countSubstrings(string s, string t) {

}
};
```

Java:

```
class Solution {
public int countSubstrings(String s, String t) {

}
}
```

Python3:

```
class Solution:
def countSubstrings(self, s: str, t: str) -> int:
```

Python:

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

JavaScript:

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

```
 * @return {number}
 */
var countSubstrings = function(s, t) {

};
```

TypeScript:

```
function countSubstrings(s: string, t: string): number {

};
```

C#:

```
public class Solution {
public int CountSubstrings(string s, string t) {

}
}
```

C:

```
int countSubstrings(char* s, char* t) {

}
```

Go:

```
func countSubstrings(s string, t string) int {

}
```

Kotlin:

```
class Solution {
fun countSubstrings(s: String, t: String): Int {

}
}
```

Swift:

```
class Solution {
func countSubstrings(_ s: String, _ t: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_substrings(s: String, t: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {String} t
# @return {Integer}
def count_substrings(s, t)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param String $t
* @return Integer
*/
function countSubstrings($s, $t) {

}
}
```

Dart:

```
class Solution {
int countSubstrings(String s, String t) {

}
```

```
}
```

Scala:

```
object Solution {
def countSubstrings(s: String, t: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_substrings(s :: String.t, t :: String.t) :: integer
def count_substrings(s, t) do

end
end
```

Erlang:

```
-spec count_substrings(S :: unicode:unicode_binary(), T ::
unicode:unicode_binary()) -> integer().
count_substrings(S, T) ->
.
```

Racket:

```
(define/contract (count-substrings s t)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Substrings That Differ by One Character
* Difficulty: Medium
* Tags: string, tree, dp, hash
```

```
*
* 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:
int countSubstrings(string s, string t) {

}
};
```

Java Solution:

```
/**
* Problem: Count Substrings That Differ by One Character
* Difficulty: Medium
* Tags: string, tree, dp, hash
*
* 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 int countSubstrings(String s, String t) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Substrings That Differ by One Character
Difficulty: Medium
Tags: string, tree, dp, hash

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 countSubstrings(self, s: str, t: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Substrings That Differ by One Character
* Difficulty: Medium
* Tags: string, tree, dp, hash
*
* 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} s
* @param {string} t
* @return {number}
*/
var countSubstrings = function(s, t) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Substrings That Differ by One Character
* Difficulty: Medium
* Tags: string, tree, dp, hash
*
* 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 countSubstrings(s: string, t: string): number {

};
```

C# Solution:

```
/*
* Problem: Count Substrings That Differ by One Character
* Difficulty: Medium
* Tags: string, tree, dp, hash
*
* 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 int CountSubstrings(string s, string t) {

}
}
```

C Solution:

```
/*
* Problem: Count Substrings That Differ by One Character
* Difficulty: Medium
* Tags: string, tree, dp, hash
*
* 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
```

```
*/

int countSubstrings(char* s, char* t) {

}
```

Go Solution:

```
// Problem: Count Substrings That Differ by One Character
// Difficulty: Medium
// Tags: string, tree, dp, hash
//
// 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 countSubstrings(s string, t string) int {

}
```

Kotlin Solution:

```
class Solution {
fun countSubstrings(s: String, t: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func countSubstrings(_ s: String, _ t: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Substrings That Differ by One Character
// Difficulty: Medium
// Tags: string, tree, dp, hash
```

```
//
// 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 count_substrings(s: String, t: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {String} t
# @return {Integer}
def count_substrings(s, t)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param String $t
* @return Integer
*/
function countSubstrings($s, $t) {

}
}
```

Dart Solution:

```
class Solution {
int countSubstrings(String s, String t) {

}
}
```

Scala Solution:

```
object Solution {
def countSubstrings(s: String, t: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_substrings(s :: String.t, t :: String.t) :: integer
def count_substrings(s, t) do

end
end
```

Erlang Solution:

```
-spec count_substrings(S :: unicode:unicode_binary(), T ::
unicode:unicode_binary()) -> integer().
count_substrings(S, T) ->
.
```

Racket Solution:

```
(define/contract (count-substrings s t)
(-> string? string? exact-integer?)
)
```