Problem 3460: Longest Common Prefix After at Most One Removal

Problem Information

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

Problem Description

You are given two strings

s

and

t

.

Return the

length

of the

longest common

prefix

between

s

and

t

after removing

at most

one character from

s

.

Note:

s

can be left without any removal.

Example 1:

Input:

s = "madxa", t = "madam"

Output:

4

Explanation:

Removing

s[3]

from

s

results in

"mada"

, which has a longest common prefix of length 4 with

t

.

Example 2:

Input:

s = "leetcode", t = "eetcode"

Output:

7

Explanation:

Removing

s[0]

from

s

results in

"eetcode"

, which matches

t

.

Example 3:

Input:

s = "one", t = "one"

Output:

3

Explanation:

No removal is needed.

Example 4:

Input:

s = "a", t = "b"

Output:

0

Explanation:

s

and

t

cannot have a common prefix.

Constraints:

1 <= s.length <= 10

5

1 <= t.length <= 10

5

s

and

t

contain only lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

```
func longestCommonPrefix(s string, t string) int {
}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (longest-common-prefix s t)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Common Prefix After at Most One Removal
* 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:
int longestCommonPrefix(string s, string t) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Common Prefix After at Most One Removal
* 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 int longestCommonPrefix(String s, String t) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Common Prefix After at Most One Removal
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 longestCommonPrefix(self, s: str, t: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Longest Common Prefix After at Most One Removal
* 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 {number}
*/
var longestCommonPrefix = function(s, t) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Common Prefix After at Most One Removal
* 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 longestCommonPrefix(s: string, t: string): number {

};
```

C# Solution:

```
/*
* Problem: Longest Common Prefix After at Most One Removal
* 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 int LongestCommonPrefix(string s, string t) {

}
}
```

C Solution:

```
/*
* Problem: Longest Common Prefix After at Most One Removal
* 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
*/

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

}
```

Go Solution:

```
// Problem: Longest Common Prefix After at Most One Removal
// 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 longestCommonPrefix(s string, t string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Longest Common Prefix After at Most One Removal
// 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 longest_common_prefix(s: String, t: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param String $t
* @return Integer
*/
function longestCommonPrefix($s, $t) {
}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (longest-common-prefix s t)
(-> string? string? exact-integer?)
)
```