Problem 2486: Append Characters to String to Make Subsequence

Problem Information

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

Problem Description

You are given two strings

s

and

t

consisting of only lowercase English letters.

Return

the minimum number of characters that need to be appended to the end of

s

so that

t

becomes a

subsequence

of

s

.

A

subsequence

is a string that can be derived from another string by deleting some or no characters without changing the order of the remaining characters.

Example 1:

Input:

s = "coaching", t = "coding"

Output:

4

Explanation:

Append the characters "ding" to the end of s so that s = "coachingding". Now, t is a subsequence of s ("

co

aching

ding

"). It can be shown that appending any 3 characters to the end of s will never make t a subsequence.

Example 2:

Input:

s = "abcde", t = "a"

Output:

0

Explanation:

t is already a subsequence of s ("

a

bcde").

Example 3:

Input:

s = "z", t = "abcde"

Output:

5

Explanation:

Append the characters "abcde" to the end of s so that s = "zabcde". Now, t is a subsequence of s ("z

abcde

"). It can be shown that appending any 4 characters to the end of s will never make t a subsequence.

Constraints:

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

5

s

and

t

consist only of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

```
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
```

```
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (append-characters s t)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Append Characters to String to Make Subsequence
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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 appendCharacters(string s, string t) {

}
};
```

Java Solution:

```
/**
* Problem: Append Characters to String to Make Subsequence
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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 appendCharacters(String s, String t) {

}
}
```

Python3 Solution:

```
"""
Problem: Append Characters to String to Make Subsequence
```

```
Difficulty: Medium
Tags: array, string, greedy

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Append Characters to String to Make Subsequence
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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 appendCharacters = function(s, t) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Append Characters to String to Make Subsequence
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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 appendCharacters(s: string, t: string): number {

};
```

C# Solution:

```
/*
* Problem: Append Characters to String to Make Subsequence
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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 AppendCharacters(string s, string t) {

}
}
```

C Solution:

```
/*
* Problem: Append Characters to String to Make Subsequence
```

```
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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 appendCharacters(char* s, char* t) {

}
```

Go Solution:

```
// Problem: Append Characters to String to Make Subsequence
// Difficulty: Medium
// Tags: array, string, greedy
//
// 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 appendCharacters(s string, t string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Append Characters to String to Make Subsequence
// Difficulty: Medium
// Tags: array, string, greedy
//
// 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 append_characters(s: String, t: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (append-characters s t)
(-> string? string? exact-integer?)
)
```