Problem 2207: Maximize Number of Subsequences in a String

Problem Information

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

Problem Description

You are given a

0-indexed

string

text

and another

0-indexed

string

pattern

of length

2

, both of which consist of only lowercase English letters.

You can add

either

pattern[0]

or

pattern[1]

anywhere in

text

exactly once

. Note that the character can be added even at the beginning or at the end of

text

.

Return

the

maximum

number of times

pattern

can occur as a

subsequence

of the modified

text

.

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:

text = "abdcdbc", pattern = "ac"

Output:

4

Explanation:

If we add pattern[0] = 'a' in between text[1] and text[2], we get "ab

a

dcdbc". Now, the number of times "ac" occurs as a subsequence is 4. Some other strings which have 4 subsequences "ac" after adding a character to text are "

a

abdcdbc" and "abd

a

cdbc". However, strings such as "abdc

a

dbc", "abd

c

cdbc", and "abdcdbc

c

", although obtainable, have only 3 subsequences "ac" and are thus suboptimal. It can be shown that it is not possible to get more than 4 subsequences "ac" by adding only one character.

Example 2:

Input:

text = "aabb", pattern = "ab"

Output:

6

Explanation:

Some of the strings which can be obtained from text and have 6 subsequences "ab" are "

a

aabb", "aa

a

bb", and "aab

b

b".

Constraints:

1 <= text.length <= 10

5

pattern.length == 2

text

and

pattern

consist only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
long long maximumSubsequenceCount(string text, string pattern) {

}
};
```

Java:

```
class Solution {
public long maximumSubsequenceCount(String text, String pattern) {

}
}
```

Python3:

```
class Solution:
def maximumSubsequenceCount(self, text: str, pattern: str) -> int:
```

Python:

```
class Solution(object):
def maximumSubsequenceCount(self, text, pattern):
"""
:type text: str
```

```
:type pattern: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} text
* @param {string} pattern
* @return {number}
*/
var maximumSubsequenceCount = function(text, pattern) {

};
```

TypeScript:

```
function maximumSubsequenceCount(text: string, pattern: string): number {

};
```

C#:

```
public class Solution {
public long MaximumSubsequenceCount(string text, string pattern) {

}
}
```

C:

```
long long maximumSubsequenceCount(char* text, char* pattern) {

}
```

Go:

```
func maximumSubsequenceCount(text string, pattern string) int64 {

}
```

Kotlin:

```
class Solution {
fun maximumSubsequenceCount(text: String, pattern: String): Long {

}
}
```

Swift:

```
class Solution {
func maximumSubsequenceCount(_ text: String, _ pattern: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn maximum_subsequence_count(text: String, pattern: String) -> i64 {

}
}
```

Ruby:

```
# @param {String} text
# @param {String} pattern
# @return {Integer}
def maximum_subsequence_count(text, pattern)

end
```

PHP:

```
class Solution {

/**
* @param String $text
* @param String $pattern
* @return Integer
*/
function maximumSubsequenceCount($text, $pattern) {

}
}
```

Dart:

```
class Solution {
int maximumSubsequenceCount(String text, String pattern) {

}
}
```

Scala:

```
object Solution {
def maximumSubsequenceCount(text: String, pattern: String): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec maximum_subsequence_count(text :: String.t, pattern :: String.t) ::
integer
def maximum_subsequence_count(text, pattern) do

end
end
```

Erlang:

```
-spec maximum_subsequence_count(Text :: unicode:unicode_binary(), Pattern ::
unicode:unicode_binary()) -> integer().
maximum_subsequence_count(Text, Pattern) ->
.
```

Racket:

```
(define/contract (maximum-subsequence-count text pattern)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximize Number of Subsequences in a String
* 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:
long long maximumSubsequenceCount(string text, string pattern) {

}
};
```

Java Solution:

```
/**
* Problem: Maximize Number of Subsequences in a String
* 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 long maximumSubsequenceCount(String text, String pattern) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximize Number of Subsequences in a String
```

```
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 maximumSubsequenceCount(self, text: str, pattern: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maximumSubsequenceCount(self, text, pattern):
"""
:type text: str
:type pattern: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximize Number of Subsequences in a String
* 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} text
* @param {string} pattern
* @return {number}
*/
var maximumSubsequenceCount = function(text, pattern) {
};
```

TypeScript Solution:

```
/**
* Problem: Maximize Number of Subsequences in a String
* 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 maximumSubsequenceCount(text: string, pattern: string): number {

};
```

C# Solution:

```
/*
* Problem: Maximize Number of Subsequences in a String
* 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 long MaximumSubsequenceCount(string text, string pattern) {

}
}
```

C Solution:

```
/*
* Problem: Maximize Number of Subsequences in a String
* 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
*/

long long maximumSubsequenceCount(char* text, char* pattern) {

}
```

Go Solution:

```
// Problem: Maximize Number of Subsequences in a String
// 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 maximumSubsequenceCount(text string, pattern string) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun maximumSubsequenceCount(text: String, pattern: String): Long {

}
}
```

Swift Solution:

```
class Solution {
func maximumSubsequenceCount(_ text: String, _ pattern: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximize Number of Subsequences in a String
// 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 maximum_subsequence_count(text: String, pattern: String) -> i64 {

}
}
```

Ruby Solution:

```
# @param {String} text
# @param {String} pattern
# @return {Integer}
def maximum_subsequence_count(text, pattern)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $text
* @param String $pattern
* @return Integer
*/
function maximumSubsequenceCount($text, $pattern) {

}
}
```

Dart Solution:

```
class Solution {
int maximumSubsequenceCount(String text, String pattern) {

}
}
```

Scala Solution:

```
object Solution {
def maximumSubsequenceCount(text: String, pattern: String): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec maximum_subsequence_count(text :: String.t, pattern :: String.t) ::
integer
def maximum_subsequence_count(text, pattern) do

end
end
```

Erlang Solution:

```
-spec maximum_subsequence_count(Text :: unicode:unicode_binary(), Pattern ::
unicode:unicode_binary()) -> integer().
maximum_subsequence_count(Text, Pattern) ->
.
```

Racket Solution:

```
(define/contract (maximum-subsequence-count text pattern)
(-> string? string? exact-integer?)
)
```