Problem 3303: Find the Occurrence of First Almost Equal Substring

Problem Information

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

Problem Description

You are given two strings

s

and

pattern

.

A string

x

is called

almost equal

to

y

if you can change

at most

one character in

x

to make it

identical

to

y

.

Return the

smallest

starting index

of a

substring

in

s

that is

almost equal

to

pattern

. If no such index exists, return

-1

.

A

substring

is a contiguous

non-empty

sequence of characters within a string.

Example 1:

Input:

s = "abcdefg", pattern = "bcdffg"

Output:

1

Explanation:

The substring

s[1..6] == "bcdefg"

can be converted to

"bcdffg"

by changing

s[4]

to

"f"

.

Example 2:

Input:

s = "ababbababa", pattern = "bacaba"

Output:

4

Explanation:

The substring

s[4..9] == "bababa"

can be converted to

"bacaba"

by changing

s[6]

to

"c"

.

Example 3:

Input:

s = "abcd", pattern = "dba"

Output:

-1

Example 4:

Input:

s = "dde", pattern = "d"

Output:

0

Constraints:

1 <= pattern.length < s.length <= 10

5

s

and

pattern

consist only of lowercase English letters.

Follow-up:

Could you solve the problem if

at most

k

consecutive

characters can be changed?

Code Snippets

C++:

```
class Solution {
public:
int minStartingIndex(string s, string pattern) {

}
};
```

Java:

```
class Solution {
public int minStartingIndex(String s, String pattern) {

}
}
```

Python3:

```
class Solution:
def minStartingIndex(self, s: str, pattern: str) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minStartingIndex(s: string, pattern: string): number {

};
```

C#:

```
public class Solution {
public int MinStartingIndex(string s, string pattern) {

}
}
```

C:

```
int minStartingIndex(char* s, char* pattern) {

}
```

Go:

```
func minStartingIndex(s string, pattern string) int {

}
```

Kotlin:

```
class Solution {
fun minStartingIndex(s: String, pattern: String): Int {

}
}
```

Swift:

```
class Solution {
func minStartingIndex(_ s: String, _ pattern: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_starting_index(s: String, pattern: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {String} pattern
# @return {Integer}
def min_starting_index(s, pattern)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param String $pattern
* @return Integer
*/
function minStartingIndex($s, $pattern) {

}
}
```

Dart:

```
class Solution {
int minStartingIndex(String s, String pattern) {

}
}
```

Scala:

```
object Solution {
def minStartingIndex(s: String, pattern: String): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec min_starting_index(S :: unicode:unicode_binary(), Pattern ::
unicode:unicode_binary()) -> integer().
min_starting_index(S, Pattern) ->
.
```

Racket:

```
(define/contract (min-starting-index s pattern)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Occurrence of First Almost Equal Substring
* Difficulty: Hard
* Tags: string, tree
```

```
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
int minStartingIndex(string s, string pattern) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Occurrence of First Almost Equal Substring
* Difficulty: Hard
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int minStartingIndex(String s, String pattern) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Occurrence of First Almost Equal Substring
Difficulty: Hard
Tags: string, tree

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
```

```
"""

class Solution:
def minStartingIndex(self, s: str, pattern: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Occurrence of First Almost Equal Substring
* Difficulty: Hard
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {string} s
* @param {string} pattern
* @return {number}
*/
var minStartingIndex = function(s, pattern) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Occurrence of First Almost Equal Substring
* Difficulty: Hard
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function minStartingIndex(s: string, pattern: string): number {

};
```

C# Solution:

```
/*
* Problem: Find the Occurrence of First Almost Equal Substring
* Difficulty: Hard
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public int MinStartingIndex(string s, string pattern) {

}
}
```

C Solution:

```
/*
* Problem: Find the Occurrence of First Almost Equal Substring
* Difficulty: Hard
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
```

```
*/

int minStartingIndex(char* s, char* pattern) {

}
```

Go Solution:

```
// Problem: Find the Occurrence of First Almost Equal Substring
// Difficulty: Hard
// Tags: string, tree
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func minStartingIndex(s string, pattern string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minStartingIndex(s: String, pattern: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func minStartingIndex(_ s: String, _ pattern: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Occurrence of First Almost Equal Substring
// Difficulty: Hard
// Tags: string, tree
```

```
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn min_starting_index(s: String, pattern: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {String} pattern
# @return {Integer}
def min_starting_index(s, pattern)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param String $pattern
* @return Integer
*/
function minStartingIndex($s, $pattern) {

}
}
```

Dart Solution:

```
class Solution {
int minStartingIndex(String s, String pattern) {

}
}
```

Scala Solution:

```
object Solution {
def minStartingIndex(s: String, pattern: String): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec min_starting_index(S :: unicode:unicode_binary(), Pattern ::
unicode:unicode_binary()) -> integer().
min_starting_index(S, Pattern) ->
.
```

Racket Solution:

```
(define/contract (min-starting-index s pattern)
(-> string? string? exact-integer?)
)
```