Problem 2565: Subsequence With the Minimum Score

Problem Information

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

Problem Description

You are given two strings

s

and

t

.

You are allowed to remove any number of characters from the string

t

.

The score of the string is

0

if no characters are removed from the string

t

, otherwise:

Let

left

be the minimum index among all removed characters.

Let

right

be the maximum index among all removed characters.

Then the score of the string is

right - left + 1

.

Return

the minimum possible score to make

t

a subsequence of

s

.

A

subsequence

of a string is a new string that is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (i.e.,

"ace"

is a subsequence of

"

a

b

c

d

e

"

while

"aec"

is not).

Example 1:

Input:

s = "abacaba", t = "bzaa"

Output:

1

Explanation:

In this example, we remove the character "z" at index 1 (0-indexed). The string t becomes "baa" which is a subsequence of the string "abacaba" and the score is 1 - 1 + 1 = 1. It can be proven that 1 is the minimum score that we can achieve.

Example 2:

Input:

s = "cde", t = "xyz"

Output:

3

Explanation:

In this example, we remove characters "x", "y" and "z" at indices 0, 1, and 2 (0-indexed). The string t becomes "" which is a subsequence of the string "cde" and the score is 2 - 0 + 1 = 3. It can be proven that 3 is the minimum score that we can achieve.

Constraints:

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

5

s

and

t

consist of only lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
end
end
```

Erlang:

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

Racket:

```
(define/contract (minimum-score s t)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Subsequence With the Minimum Score
* Difficulty: Hard
* Tags: array, string, search
*
* 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 minimumScore(string s, string t) {

}
};
```

Java Solution:

```
/**
* Problem: Subsequence With the Minimum Score
* Difficulty: Hard
* Tags: array, string, search
*
* 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 minimumScore(String s, String t) {

}
}
```

Python3 Solution:

```
"""
Problem: Subsequence With the Minimum Score
Difficulty: Hard
Tags: array, string, search

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Subsequence With the Minimum Score
* Difficulty: Hard
* Tags: array, string, search
*
* 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 minimumScore = function(s, t) {

};
```

TypeScript Solution:

```
/**
* Problem: Subsequence With the Minimum Score
* Difficulty: Hard
* Tags: array, string, search
*
* 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 minimumScore(s: string, t: string): number {

};
```

C# Solution:

```
/*
* Problem: Subsequence With the Minimum Score
* Difficulty: Hard
```

```
* Tags: array, string, search
*
* 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 MinimumScore(string s, string t) {

}
}
```

C Solution:

```
/*
* Problem: Subsequence With the Minimum Score
* Difficulty: Hard
* Tags: array, string, search
*
* 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 minimumScore(char* s, char* t) {

}
```

Go Solution:

```
// Problem: Subsequence With the Minimum Score
// Difficulty: Hard
// Tags: array, string, search
//
// 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 minimumScore(s string, t string) int {
```

```
}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Subsequence With the Minimum Score
// Difficulty: Hard
// Tags: array, string, search
//
// 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 minimum_score(s: String, t: String) -> i32 {

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (minimum-score s t)
(-> string? string? exact-integer?)
)
```