Problem 522: Longest Uncommon Subsequence II

Problem Information

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

Problem Description

Given an array of strings

strs

, return

the length of the

longest uncommon subsequence

between them

. If the longest uncommon subsequence does not exist, return

-1

.

An

uncommon subsequence

between an array of strings is a string that is a

subsequence of one string but not the others

.

A

subsequence

of a string

s

is a string that can be obtained after deleting any number of characters from

s

.

For example,

"abc"

is a subsequence of

"aebdc"

because you can delete the underlined characters in

"a

e

b

d

c"

to get

"abc"

. Other subsequences of

"aebdc"

include

"aebdc"

,

"aeb"

, and

""

(empty string).

Example 1:

Input:

strs = ["aba","cdc","eae"]

Output:

3

Example 2:

Input:

strs = ["aaa","aaa","aa"]

Output:

-1

Constraints:

2 <= strs.length <= 50

1 <= strs[i].length <= 10

strs[i]

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int findLUSlength(vector<string>& strs) {

}
};
```

Java:

```
class Solution {
public int findLUSlength(String[] strs) {

}
}
```

Python3:

```
class Solution:
def findLUSlength(self, strs: List[str]) -> int:
```

Python:

```
class Solution(object):
def findLUSlength(self, strs):
"""
:type strs: List[str]
```

```
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string[]} strs
* @return {number}
*/
var findLUSlength = function(strs) {

};
```

TypeScript:

```
function findLUSlength(strs: string[]): number {

};
```

C#:

```
public class Solution {
public int FindLUSlength(string[] strs) {

}
}
```

C:

```
int findLUSlength(char** strs, int strsSize) {

}
```

Go:

```
func findLUSlength(strs []string) int {

}
```

Kotlin:

```
class Solution {
fun findLUSlength(strs: Array<String>): Int {

}
}
```

Swift:

```
class Solution {
func findLUSlength(_ strs: [String]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn find_lu_slength(strs: Vec<String>) -> i32 {

}
}
```

Ruby:

```
# @param {String[]} strs
# @return {Integer}
def find_lu_slength(strs)

end
```

PHP:

```
class Solution {

/**
* @param String[] $strs
* @return Integer
*/
function findLUSlength($strs) {

}
}
```

Dart:

```
class Solution {
int findLUSlength(List<String> strs) {

}
}
```

Scala:

```
object Solution {
def findLUSlength(strs: Array[String]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_lu_slength(strs :: [String.t]) :: integer
def find_lu_slength(strs) do

end
end
```

Erlang:

```
-spec find_lu_slength(Strs :: [unicode:unicode_binary()]) -> integer().
find_lu_slength(Strs) ->
.
```

Racket:

```
(define/contract (find-lu-slength strs)
(-> (listof string?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Uncommon Subsequence II
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int findLUSlength(vector<string>& strs) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Uncommon Subsequence II
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int findLUSlength(String[] strs) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Uncommon Subsequence II
Difficulty: Medium
Tags: array, string, hash, sort
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def findLUSlength(self, strs: List[str]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def findLUSlength(self, strs):
"""
:type strs: List[str]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Longest Uncommon Subsequence II
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string[]} strs
* @return {number}
*/
var findLUSlength = function(strs) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Uncommon Subsequence II
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function findLUSlength(strs: string[]): number {

};
```

C# Solution:

```
/*
* Problem: Longest Uncommon Subsequence II
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int FindLUSlength(string[] strs) {

}
}
```

C Solution:

```
/*
* Problem: Longest Uncommon Subsequence II
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

int findLUSlength(char** strs, int strsSize) {

}
```

Go Solution:

```
// Problem: Longest Uncommon Subsequence II
// Difficulty: Medium
// Tags: array, string, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func findLUSlength(strs []string) int {

}
```

Kotlin Solution:

```
class Solution {
fun findLUSlength(strs: Array<String>): Int {

}
}
```

Swift Solution:

```
class Solution {
func findLUSlength(_ strs: [String]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Longest Uncommon Subsequence II
// Difficulty: Medium
// Tags: array, string, hash, sort
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn find_lu_slength(strs: Vec<String>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String[]} strs
# @return {Integer}
def find_lu_slength(strs)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $strs
* @return Integer
*/
function findLUSlength($strs) {

}
}
```

Dart Solution:

```
class Solution {
int findLUSlength(List<String> strs) {

}
}
```

Scala Solution:

```
object Solution {
def findLUSlength(strs: Array[String]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_lu_slength(strs :: [String.t]) :: integer
def find_lu_slength(strs) do

end
end
```

Erlang Solution:

```
-spec find_lu_slength(Strs :: [unicode:unicode_binary()]) -> integer().
find_lu_slength(Strs) ->
.
```

Racket Solution:

```
(define/contract (find-lu-slength strs)
(-> (listof string?) exact-integer?)
)
```