Problem 2707: Extra Characters in a String

Problem Information

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

Problem Description

You are given a

0-indexed

string

s

and a dictionary of words

dictionary

. You have to break

s

into one or more

non-overlapping

substrings such that each substring is present in

dictionary

. There may be some

extra characters

in

s

which are not present in any of the substrings.

Return

the

minimum

number of extra characters left over if you break up

s

optimally.

Example 1:

Input:

s = "leetscode", dictionary = ["leet","code","leetcode"]

Output:

1

Explanation:

We can break s in two substrings: "leet" from index 0 to 3 and "code" from index 5 to 8. There is only 1 unused character (at index 4), so we return 1.

Example 2:

Input:

s = "sayhelloworld", dictionary = ["hello","world"]

Output:

3

Explanation:

We can break s in two substrings: "hello" from index 3 to 7 and "world" from index 8 to 12. The characters at indices 0, 1, 2 are not used in any substring and thus are considered as extra characters. Hence, we return 3.

Constraints:

1 <= s.length <= 50

1 <= dictionary.length <= 50

1 <= dictionary[i].length <= 50

dictionary[i]

and

s

consists of only lowercase English letters

dictionary

contains distinct words

Code Snippets

C++:

```
class Solution {
public:
```

```
int minExtraChar(string s, vector<string>& dictionary) {

}
};
```

Java:

```
class Solution {
public int minExtraChar(String s, String[] dictionary) {

}
}
```

Python3:

```
class Solution:
def minExtraChar(self, s: str, dictionary: List[str]) -> int:
```

Python:

```
class Solution(object):
def minExtraChar(self, s, dictionary):
"""
:type s: str
:type dictionary: List[str]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {string[]} dictionary
* @return {number}
*/
var minExtraChar = function(s, dictionary) {

};
```

TypeScript:

```
function minExtraChar(s: string, dictionary: string[]): number {

};
```

C#:

```
public class Solution {
public int MinExtraChar(string s, string[] dictionary) {

}
}
```

C:

```
int minExtraChar(char* s, char** dictionary, int dictionarySize) {

}
```

Go:

```
func minExtraChar(s string, dictionary []string) int {

}
```

Kotlin:

```
class Solution {
fun minExtraChar(s: String, dictionary: Array<String>): Int {

}
}
```

Swift:

```
class Solution {
func minExtraChar(_ s: String, _ dictionary: [String]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_extra_char(s: String, dictionary: Vec<String>) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {String[]} dictionary
# @return {Integer}
def min_extra_char(s, dictionary)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param String[] $dictionary
* @return Integer
*/
function minExtraChar($s, $dictionary) {

}
}
```

Dart:

```
class Solution {
int minExtraChar(String s, List<String> dictionary) {

}
}
```

Scala:

```
object Solution {
def minExtraChar(s: String, dictionary: Array[String]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_extra_char(s :: String.t, dictionary :: [String.t]) :: integer
def min_extra_char(s, dictionary) do

end
end
```

Erlang:

```
-spec min_extra_char(S :: unicode:unicode_binary(), Dictionary ::
[unicode:unicode_binary()]) -> integer().
min_extra_char(S, Dictionary) ->
.
```

Racket:

```
(define/contract (min-extra-char s dictionary)
(-> string? (listof string?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Extra Characters in a String
* Difficulty: Medium
* Tags: array, string, tree, dp, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
```

```
int minExtraChar(string s, vector<string>& dictionary) {

}
};
```

Java Solution:

```
/**
* Problem: Extra Characters in a String
* Difficulty: Medium
* Tags: array, string, tree, dp, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int minExtraChar(String s, String[] dictionary) {

}
}
```

Python3 Solution:

```
"""
Problem: Extra Characters in a String
Difficulty: Medium
Tags: array, string, tree, dp, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

```
class Solution(object):
def minExtraChar(self, s, dictionary):
"""
:type s: str
:type dictionary: List[str]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Extra Characters in a String
* Difficulty: Medium
* Tags: array, string, tree, dp, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} s
* @param {string[]} dictionary
* @return {number}
*/
var minExtraChar = function(s, dictionary) {

};
```

TypeScript Solution:

```
/**
* Problem: Extra Characters in a String
* Difficulty: Medium
* Tags: array, string, tree, dp, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function minExtraChar(s: string, dictionary: string[]): number {
};
```

C# Solution:

```
/*
* Problem: Extra Characters in a String
* Difficulty: Medium
* Tags: array, string, tree, dp, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int MinExtraChar(string s, string[] dictionary) {

}
}
```

C Solution:

```
/*
* Problem: Extra Characters in a String
* Difficulty: Medium
* Tags: array, string, tree, dp, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int minExtraChar(char* s, char** dictionary, int dictionarySize) {

}
```

Go Solution:

```
// Problem: Extra Characters in a String
// Difficulty: Medium
// Tags: array, string, tree, dp, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func minExtraChar(s string, dictionary []string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minExtraChar(s: String, dictionary: Array<String>): Int {

}
}
```

Swift Solution:

```
class Solution {
func minExtraChar(_ s: String, _ dictionary: [String]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Extra Characters in a String
// Difficulty: Medium
// Tags: array, string, tree, dp, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn min_extra_char(s: String, dictionary: Vec<String>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {String[]} dictionary
# @return {Integer}
def min_extra_char(s, dictionary)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param String[] $dictionary
* @return Integer
*/
function minExtraChar($s, $dictionary) {

}
}
```

Dart Solution:

```
class Solution {
int minExtraChar(String s, List<String> dictionary) {

}
}
```

Scala Solution:

```
object Solution {
def minExtraChar(s: String, dictionary: Array[String]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_extra_char(s :: String.t, dictionary :: [String.t]) :: integer
def min_extra_char(s, dictionary) do

end
end
```

Erlang Solution:

```
-spec min_extra_char(S :: unicode:unicode_binary(), Dictionary ::
[unicode:unicode_binary()]) -> integer().
min_extra_char(S, Dictionary) ->
.
```

Racket Solution:

```
(define/contract (min-extra-char s dictionary)
(-> string? (listof string?) exact-integer?)
)
```