Problem 2430: Maximum Deletions on a String

Problem Information

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

Problem Description

You are given a string

s

consisting of only lowercase English letters. In one operation, you can:

Delete

the entire string

s

, or

Delete the

first

i

letters of

s

if the first

i

letters of

s

are

equal

to the following

i

letters in

s

, for any

i

in the range

1 <= i <= s.length / 2

.

For example, if

s = "ababc"

, then in one operation, you could delete the first two letters of

s

to get

"abc"

, since the first two letters of

s

and the following two letters of

s

are both equal to

"ab"

.

Return

the

maximum

number of operations needed to delete all of

s

.

Example 1:

Input:

s = "abcabcdabc"

Output:

2

Explanation:

- Delete the first 3 letters ("abc") since the next 3 letters are equal. Now, s = "abcdabc". - Delete all the letters. We used 2 operations so return 2. It can be proven that 2 is the maximum number of operations needed. Note that in the second operation we cannot delete "abc" again because the next occurrence of "abc" does not happen in the next 3 letters.

Example 2:

Input:

s = "aaabaab"

Output:

4

Explanation:

- Delete the first letter ("a") since the next letter is equal. Now, s = "aabaab". - Delete the first 3 letters ("aab") since the next 3 letters are equal. Now, s = "aab". - Delete the first letter ("a") since the next letter is equal. Now, s = "ab". - Delete all the letters. We used 4 operations so return 4. It can be proven that 4 is the maximum number of operations needed.

Example 3:

Input:

s = "aaaaa"

Output:

5

Explanation:

In each operation, we can delete the first letter of s.

Constraints:

1 <= s.length <= 4000

`s`

consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int deleteString(string s) {

}
};
```

Java:

```
class Solution {
public int deleteString(String s) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {string} s
```

```
* @return {number}
*/
var deleteString = function(s) {

};
```

TypeScript:

```
function deleteString(s: string): number {

};
```

C#:

```
public class Solution {
public int DeleteString(string s) {

}
}
```

C:

```
int deleteString(char* s) {

}
```

Go:

```
func deleteString(s string) int {

}
```

Kotlin:

```
class Solution {
fun deleteString(s: String): Int {

}
}
```

Swift:

```
class Solution {
func deleteString(_ s: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn delete_string(s: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @return {Integer}
def delete_string(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function deleteString($s) {

}
}
```

Dart:

```
class Solution {
int deleteString(String s) {

}
}
```

Scala:

```
object Solution {
def deleteString(s: String): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec delete_string(S :: unicode:unicode_binary()) -> integer().
delete_string(S) ->
.
```

Racket:

```
(define/contract (delete-string s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Deletions on a String
* Difficulty: Hard
* Tags: string, dp, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

```
class Solution {
public:
int deleteString(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Deletions on a String
* Difficulty: Hard
* Tags: string, dp, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int deleteString(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Deletions on a String
Difficulty: Hard
Tags: string, dp, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Deletions on a String
* Difficulty: Hard
* Tags: string, dp, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} s
* @return {number}
*/
var deleteString = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Deletions on a String
* Difficulty: Hard
* Tags: string, dp, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

function deleteString(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Deletions on a String
* Difficulty: Hard
* Tags: string, dp, hash
*
* Approach: String manipulation with hash map or two pointers
* 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 DeleteString(string s) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Deletions on a String
* Difficulty: Hard
* Tags: string, dp, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int deleteString(char* s) {

}
```

Go Solution:

```
// Problem: Maximum Deletions on a String
// Difficulty: Hard
// Tags: string, dp, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func deleteString(s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun deleteString(s: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func deleteString(_ s: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Deletions on a String
// Difficulty: Hard
// Tags: string, dp, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn delete_string(s: String) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {String} s
# @return {Integer}
def delete_string(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function deleteString($s) {

}
}
```

Dart Solution:

```
class Solution {
int deleteString(String s) {

}
}
```

Scala Solution:

```
object Solution {
def deleteString(s: String): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec delete_string(S :: unicode:unicode_binary()) -> integer().
delete_string(S) ->
.
```

Racket Solution:

```
(define/contract (delete-string s)
(-> string? exact-integer?)
)
```