Problem 2539: Count the Number of Good Subsequences

Problem Information

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

Problem Description

A

subsequence

of a string is good if it is not empty and the frequency of each one of its characters is the same.

Given a string

s

, return

the number of good subsequences of

s

. Since the answer may be too large, return it modulo

10

9

+ 7

.

A

subsequence

is a string that can be derived from another string by deleting some or no characters without changing the order of the remaining characters.

Example 1:

Input:

s = "aabb"

Output:

11

Explanation:

The total number of subsequences is

2

4

.

There are five subsequences which are not good: "

aab

b", "a

abb

", "

a

a

bb

", "

aa

b

b

", and the empty subsequence. Hence, the number of good subsequences is

2

4

-5 = 11

.

Example 2:

Input:

s = "leet"

Output:

12

Explanation:

There are four subsequences which are not good: "

l

ee

t", "l

eet

", "

leet

", and the empty subsequence. Hence, the number of good subsequences is

2

4

-4 = 12

.

Example 3:

Input:

s = "abcd"

Output:

15

Explanation:

All of the non-empty subsequences are good subsequences. Hence, the number of good subsequences is

2

4

-1 = 15

.

Constraints:

1 <= s.length <= 10

4

s

consists of only lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int countGoodSubsequences(char* s) {

}
```

Go:

```
func countGoodSubsequences(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (count-good-subsequences s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count the Number of Good Subsequences
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int countGoodSubsequences(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Count the Number of Good Subsequences
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int countGoodSubsequences(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Count the Number of Good Subsequences
Difficulty: Medium
Tags: string, math, hash
```

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

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count the Number of Good Subsequences
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Count the Number of Good Subsequences
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function countGoodSubsequences(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Count the Number of Good Subsequences
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int CountGoodSubsequences(string s) {

}
}
```

C Solution:

```
/*
* Problem: Count the Number of Good Subsequences
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

int countGoodSubsequences(char* s) {

}
```

Go Solution:

```
// Problem: Count the Number of Good Subsequences
// Difficulty: Medium
// Tags: string, math, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func countGoodSubsequences(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Count the Number of Good Subsequences
// Difficulty: Medium
// Tags: string, math, hash
```

```
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (count-good-subsequences s)
(-> string? exact-integer?)
)
```