Problem 2930: Number of Strings Which Can Be Rearranged to Contain Substring

Problem Information

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

Problem Description

You are given an integer

n

.

A string

s

is called

good

if it contains only lowercase English characters

and

it is possible to rearrange the characters of

s

such that the new string contains

"leet"

as a

substring

.

For example:

The string

"lteer"

is good because we can rearrange it to form

"leetr"

.

"letl"

is not good because we cannot rearrange it to contain

"leet"

as a substring.

Return

the

total

number of good strings of length

n

.

Since the answer may be large, return it

modulo

10

9

+ 7

.

A

substring

is a contiguous sequence of characters within a string.

Example 1:

Input:

n = 4

Output:

12

Explanation:

The 12 strings which can be rearranged to have "leet" as a substring are: "eelt", "eetl", "elet", "elte", "etel", "etle", "leet", "lete", "ltee", "teel", "tele", and "tlee".

Example 2:

Input:

n = 10

Output:

83943898

Explanation:

The number of strings with length 10 which can be rearranged to have "leet" as a substring is 526083947580. Hence the answer is 526083947580 % (10

9

+ 7) = 83943898.

Constraints:

1 <= n <= 10

5

Code Snippets

C++:

```
class Solution {
public:
int stringCount(int n) {

}
};
```

Java:

```
class Solution {
public int stringCount(int n) {

}
}
```

Python3:

```
class Solution:
def stringCount(self, n: int) -> int:
```

Python:

```
class Solution(object):
def stringCount(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {number}
*/
var stringCount = function(n) {

};
```

TypeScript:

```
function stringCount(n: number): number {

};
```

C#:

```
public class Solution {
public int StringCount(int n) {

}
}
```

C:

```
int stringCount(int n) {

}
```

Go:

```
func stringCount(n int) int {

}
```

Kotlin:

```
class Solution {
fun stringCount(n: Int): Int {

}
}
```

Swift:

```
class Solution {
func stringCount(_ n: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn string_count(n: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @return {Integer}
def string_count(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function stringCount($n) {

}
}
```

Dart:

```
class Solution {
int stringCount(int n) {

}
}
```

Scala:

```
object Solution {
def stringCount(n: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec string_count(n :: integer) :: integer
def string_count(n) do

end
end
```

Erlang:

```
-spec string_count(N :: integer()) -> integer().
string_count(N) ->
.
```

Racket:

```
(define/contract (string-count n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Strings Which Can Be Rearranged to Contain Substring
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* 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 stringCount(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Strings Which Can Be Rearranged to Contain Substring
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* 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 stringCount(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Strings Which Can Be Rearranged to Contain Substring
Difficulty: Medium
Tags: string, tree, dp, math

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 stringCount(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def stringCount(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Strings Which Can Be Rearranged to Contain Substring
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* 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 {number} n
* @return {number}
*/
var stringCount = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Strings Which Can Be Rearranged to Contain Substring
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* 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 stringCount(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Number of Strings Which Can Be Rearranged to Contain Substring
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* 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 StringCount(int n) {

}
}
```

C Solution:

```
/*
* Problem: Number of Strings Which Can Be Rearranged to Contain Substring
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* 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 stringCount(int n) {

}
```

Go Solution:

```
// Problem: Number of Strings Which Can Be Rearranged to Contain Substring
// Difficulty: Medium
// Tags: string, tree, dp, math
//
// 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 stringCount(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun stringCount(n: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func stringCount(_ n: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Strings Which Can Be Rearranged to Contain Substring
// Difficulty: Medium
// Tags: string, tree, dp, math
//
// 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 string_count(n: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Integer}
def string_count(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function stringCount($n) {

}
}
```

Dart Solution:

```
class Solution {
int stringCount(int n) {

}
}
```

Scala Solution:

```
object Solution {
def stringCount(n: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec string_count(n :: integer) :: integer
def string_count(n) do

end
end
```

Erlang Solution:

```
-spec string_count(N :: integer()) -> integer().
string_count(N) ->
.
```

Racket Solution:

```
(define/contract (string-count n)
(-> exact-integer? exact-integer?)
)
```