Problem 1513: Number of Substrings With Only 1s

Problem Information

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

Problem Description

Given a binary string

s

, return

the number of substrings with all characters

1

's

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

10

9

+ 7

.

Example 1:

Input:

s = "0110111"

Output:

9

Explanation:

There are 9 substring in total with only 1's characters. "1" -> 5 times. "11" -> 3 times. "111" -> 1 time.

Example 2:

Input:

s = "101"

Output:

2

Explanation:

Substring "1" is shown 2 times in s.

Example 3:

Input:

s = "111111"

Output:

21

Explanation:

Each substring contains only 1's characters.

Constraints:

1 <= s.length <= 10

5

s[i]

is either

'0'

or

'1'

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int numSub(char* s) {

}
```

Go:

```
func numSub(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Integer
```

```
*/
function numSub($s) {

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Number of Substrings With Only 1s
* Difficulty: Medium
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Number of Substrings With Only 1s
* Difficulty: Medium
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int numSub(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Substrings With Only 1s
Difficulty: Medium
Tags: string, tree, math

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number of Substrings With Only 1s
* Difficulty: Medium
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

```
};
```

TypeScript Solution:

```
/**
* Problem: Number of Substrings With Only 1s
* Difficulty: Medium
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function numSub(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Number of Substrings With Only 1s
* Difficulty: Medium
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
}
```

C Solution:

```
/*
* Problem: Number of Substrings With Only 1s
* Difficulty: Medium
```

```
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

int numSub(char* s) {

}
```

Go Solution:

```
// Problem: Number of Substrings With Only 1s
// Difficulty: Medium
// Tags: string, tree, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func numSub(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Number of Substrings With Only 1s
// Difficulty: Medium
// Tags: string, tree, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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