Problem 1987: Number of Unique Good Subsequences

Problem Information

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

Problem Description

You are given a binary string

binary

. A

subsequence

of

binary

is considered

good

if it is

not empty

and has

no leading zeros

(with the exception of

"0"

).

Find the number of

unique good subsequences

of

binary

.

For example, if

binary = "001"

, then all the

good

subsequences are

["0", "0", "1"]

, so the

unique

good subsequences are

"0"

and

"1"

. Note that subsequences

"00"

,

"01"

, and

"001"

are not good because they have leading zeros.

Return

the number of

unique good subsequences

of

binary

. Since the answer may be very large, return it

modulo

10

9

+ 7

.

A

subsequence

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

Example 1:

Input:

binary = "001"

Output:

2

Explanation:

The good subsequences of binary are ["0", "0", "1"]. The unique good subsequences are "0" and "1".

Example 2:

Input:

binary = "11"

Output:

2

Explanation:

The good subsequences of binary are ["1", "1", "11"]. The unique good subsequences are "1" and "11".

Example 3:

Input:

binary = "101"

Output:

5

Explanation:

The good subsequences of binary are ["1", "0", "1", "10", "11", "101"]. The unique good subsequences are "0", "1", "10", "11", and "101".

Constraints:

1 <= binary.length <= 10

5

binary

consists of only

'0'

s and

'1'

s.

Code Snippets

C++:

```
class Solution {
public:
int numberOfUniqueGoodSubsequences(string binary) {

}
};
```

Java:

```
class Solution {
public int numberOfUniqueGoodSubsequences(String binary) {

}
}
```

Python3:

```
class Solution:
def numberOfUniqueGoodSubsequences(self, binary: str) -> int:
```

Python:

```
class Solution(object):
def numberOfUniqueGoodSubsequences(self, binary):
"""
:type binary: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} binary
* @return {number}
*/
var numberOfUniqueGoodSubsequences = function(binary) {

};
```

TypeScript:

```
function numberOfUniqueGoodSubsequences(binary: string): number {

};
```

C#:

```
public class Solution {
public int NumberOfUniqueGoodSubsequences(string binary) {

}
}
```

C:

```
int numberOfUniqueGoodSubsequences(char* binary) {

}
```

Go:

```
func numberOfUniqueGoodSubsequences(binary string) int {

}
```

Kotlin:

```
class Solution {
fun numberOfUniqueGoodSubsequences(binary: String): Int {

}
}
```

Swift:

```
class Solution {
func numberOfUniqueGoodSubsequences(_ binary: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_unique_good_subsequences(binary: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} binary
# @return {Integer}
def number_of_unique_good_subsequences(binary)

end
```

PHP:

```
class Solution {

/**
* @param String $binary
* @return Integer
*/
function numberOfUniqueGoodSubsequences($binary) {

}
}
```

Dart:

```
class Solution {
int numberOfUniqueGoodSubsequences(String binary) {

}
}
```

Scala:

```
object Solution {
def numberOfUniqueGoodSubsequences(binary: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_unique_good_subsequences(binary :: String.t) :: integer
def number_of_unique_good_subsequences(binary) do

end
end
```

Erlang:

```
-spec number_of_unique_good_subsequences(Binary :: unicode:unicode_binary())
-> integer().
number_of_unique_good_subsequences(Binary) ->
```

```
.
```

Racket:

```
(define/contract (number-of-unique-good-subsequences binary)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Unique Good Subsequences
* Difficulty: Hard
* Tags: string, dp
*
* 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 numberOfUniqueGoodSubsequences(string binary) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Unique Good Subsequences
* Difficulty: Hard
* Tags: string, dp
*
* 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 numberOfUniqueGoodSubsequences(String binary) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Unique Good Subsequences
Difficulty: Hard
Tags: string, dp

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 numberOfUniqueGoodSubsequences(self, binary: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numberOfUniqueGoodSubsequences(self, binary):
"""
:type binary: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Unique Good Subsequences
* Difficulty: Hard
* Tags: string, dp
*
* 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} binary
* @return {number}
*/
var numberOfUniqueGoodSubsequences = function(binary) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Unique Good Subsequences
* Difficulty: Hard
* Tags: string, dp
*
* 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 numberOfUniqueGoodSubsequences(binary: string): number {

};
```

C# Solution:

```
/*
* Problem: Number of Unique Good Subsequences
* Difficulty: Hard
* Tags: string, dp
*
* 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 NumberOfUniqueGoodSubsequences(string binary) {

}
}
```

C Solution:

```
/*
* Problem: Number of Unique Good Subsequences
* Difficulty: Hard
* Tags: string, dp
*
* 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 numberOfUniqueGoodSubsequences(char* binary) {

}
```

Go Solution:

```
// Problem: Number of Unique Good Subsequences
// Difficulty: Hard
// Tags: string, dp
//
// 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 numberOfUniqueGoodSubsequences(binary string) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfUniqueGoodSubsequences(binary: String): Int {

}
```

```
}
```

Swift Solution:

```
class Solution {
func numberOfUniqueGoodSubsequences(_ binary: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Unique Good Subsequences
// Difficulty: Hard
// Tags: string, dp
//
// 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 number_of_unique_good_subsequences(binary: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} binary
# @return {Integer}
def number_of_unique_good_subsequences(binary)

end
```

PHP Solution:

```
class Solution {

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

```
*/
function numberOfUniqueGoodSubsequences($binary) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfUniqueGoodSubsequences(String binary) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfUniqueGoodSubsequences(binary: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_unique_good_subsequences(binary :: String.t) :: integer
def number_of_unique_good_subsequences(binary) do

end
end
```

Erlang Solution:

```
-spec number_of_unique_good_subsequences(Binary :: unicode:unicode_binary())
-> integer().
number_of_unique_good_subsequences(Binary) ->
.
```

Racket Solution:

```
(define/contract (number-of-unique-good-subsequences binary)
(-> string? exact-integer?)
)
```