Problem 1922: Count Good Numbers

Problem Information

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

Problem Description

A digit string is

good

if the digits

(0-indexed)

at

even

indices are

even

and the digits at

odd

indices are

prime

(

2

,

3

,

5

, or

7

).

For example,

"2582"

is good because the digits (

2

and

8

) at even positions are even and the digits (

5

and

2

) at odd positions are prime. However,

"3245"

is

not

good because

3

is at an even index but is not even.

Given an integer

n

, return

the

total

number of good digit strings of length

n

. Since the answer may be large,

return it modulo

10

9

+ 7

.

A

digit string

is a string consisting of digits

0

through

9

that may contain leading zeros.

Example 1:

Input:

n = 1

Output:

5

Explanation:

The good numbers of length 1 are "0", "2", "4", "6", "8".

Example 2:

Input:

n = 4

Output:

400

Example 3:

Input:

n = 50

Output:

564908303

Constraints:

1 <= n <= 10

15

Code Snippets

C++:

```
class Solution {
public:
int countGoodNumbers(long long n) {

}
};
```

Java:

```
class Solution {
public int countGoodNumbers(long n) {

}
}
```

Python3:

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

Python:

```
class Solution(object):
def countGoodNumbers(self, n):
```

```
"""
:type n: int
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public int CountGoodNumbers(long n) {

}
}
```

C:

```
int countGoodNumbers(long long n) {

}
```

Go:

```
func countGoodNumbers(n int64) int {

}
```

Kotlin:

```
class Solution {
fun countGoodNumbers(n: Long): Int {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn count_good_numbers(n: i64) -> i32 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def countGoodNumbers(n: Long): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Count Good Numbers
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int countGoodNumbers(long long n) {

}
};
```

Java Solution:

```
/**
* Problem: Count Good Numbers
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int countGoodNumbers(long n) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Good Numbers
Difficulty: Medium
Tags: string, math
```

```
Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def countGoodNumbers(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Good Numbers
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Count Good Numbers
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function countGoodNumbers(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Count Good Numbers
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int CountGoodNumbers(long n) {

}
}
```

C Solution:

```
/*
* Problem: Count Good Numbers
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

int countGoodNumbers(long long n) {

}
```

Go Solution:

```
// Problem: Count Good Numbers
// Difficulty: Medium
// Tags: string, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func countGoodNumbers(n int64) int {

}
```

Kotlin Solution:

```
class Solution {
fun countGoodNumbers(n: Long): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Count Good Numbers
// Difficulty: Medium
// Tags: string, math
```

```
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn count_good_numbers(n: i64) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

```
object Solution {
def countGoodNumbers(n: Long): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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