Problem 1416: Restore The Array

Problem Information

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

Problem Description

A program was supposed to print an array of integers. The program forgot to print whitespaces and the array is printed as a string of digits

s

and all we know is that all integers in the array were in the range

[1, k]

and there are no leading zeros in the array.

Given the string

s

and the integer

k

, return

the number of the possible arrays that can be printed as

s

using the mentioned program

. Since the answer may be very large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

s = "1000", k = 10000

Output:

1

Explanation:

The only possible array is [1000]

Example 2:

Input:

s = "1000", k = 10

Output:

0

Explanation:

There cannot be an array that was printed this way and has all integer >= 1 and <= 10.

Example 3:

Input:

s = "1317", k = 2000

Output:

8

Explanation:

Possible arrays are [1317],[131,7],[13,17],[1,317],[13,1,7],[1,31,7],[1,3,17],[1,3,1,7]

Constraints:

1 <= s.length <= 10

5

s

consists of only digits and does not contain leading zeros.

1 <= k <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int numberOfArrays(string s, int k) {
```

```
}
};
```

Java:

```
class Solution {
public int numberOfArrays(String s, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function numberOfArrays(s: string, k: number): number {

};
```

C#:

```
public class Solution {
public int NumberOfArrays(string s, int k) {

}
}
```

C:

```
int numberOfArrays(char* s, int k) {

}
```

Go:

```
func numberOfArrays(s string, k int) int {

}
```

Kotlin:

```
class Solution {
fun numberOfArrays(s: String, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func numberOfArrays(_ s: String, _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_arrays(s: String, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} k
# @return {Integer}
def number_of_arrays(s, k)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return Integer
*/
function numberOfArrays($s, $k) {

}
}
```

Dart:

```
class Solution {
int numberOfArrays(String s, int k) {

}
}
```

Scala:

```
object Solution {
def numberOfArrays(s: String, k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_arrays(s :: String.t, k :: integer) :: integer
```

```
def number_of_arrays(s, k) do

end
end
```

Erlang:

```
-spec number_of_arrays(S :: unicode:unicode_binary(), K :: integer()) ->
integer().
number_of_arrays(S, K) ->
.
```

Racket:

```
(define/contract (number-of-arrays s k)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Restore The Array
* Difficulty: Hard
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int numberOfArrays(string s, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Restore The Array
* Difficulty: Hard
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int numberOfArrays(String s, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Restore The Array
Difficulty: Hard
Tags: array, string, dp

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Restore The Array
* Difficulty: Hard
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} s
* @param {number} k
* @return {number}
*/
var numberOfArrays = function(s, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Restore The Array
* Difficulty: Hard
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function numberOfArrays(s: string, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Restore The Array
* Difficulty: Hard
```

```
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* 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 NumberOfArrays(string s, int k) {

}
}
```

C Solution:

```
/*
* Problem: Restore The Array
* Difficulty: Hard
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int numberOfArrays(char* s, int k) {

}
```

Go Solution:

```
// Problem: Restore The Array
// Difficulty: Hard
// Tags: array, string, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func numberOfArrays(s string, k int) int {
```

```
}
```

Kotlin Solution:

```
class Solution {
fun numberOfArrays(s: String, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfArrays(_ s: String, _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Restore The Array
// Difficulty: Hard
// Tags: array, string, dp
//
// Approach: Use two pointers or sliding window technique
// 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_arrays(s: String, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} k
# @return {Integer}
def number_of_arrays(s, k)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return Integer
*/
function numberOfArrays($s, $k) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfArrays(String s, int k) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfArrays(s: String, k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_arrays(s :: String.t, k :: integer) :: integer
def number_of_arrays(s, k) do

end
end
```

Erlang Solution:

```
-spec number_of_arrays(S :: unicode:unicode_binary(), K :: integer()) ->
integer().
number_of_arrays(S, K) ->
.
```

Racket Solution:

```
(define/contract (number-of-arrays s k)
(-> string? exact-integer? exact-integer?)
)
```