Problem 3448: Count Substrings Divisible By Last Digit

Problem Information

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

Problem Description

You are given a string

s

consisting of digits.

Return the

number

of

substrings

of

s

divisible

by their

non-zero

last digit.

Note

: A substring may contain leading zeros.

Example 1:

Input:

s = "12936"

Output:

11

Explanation:

Substrings

"29"

,

"129"

,

"293"

and

"2936"

are not divisible by their last digit. There are 15 substrings in total, so the answer is

15 - 4 = 11

.

Example 2:

Input:

s = "5701283"

Output:

18

Explanation:

Substrings

"01"

,

"12"

,

"701"

,

"012"

,

"128"

,

"5701"

,

"7012"

,

"0128"

,

"57012"

,

"70128"

,

"570128"

, and

"701283"

are all divisible by their last digit. Additionally, all substrings that are just 1 non-zero digit are divisible by themselves. Since there are 6 such digits, the answer is

12 + 6 = 18

.

Example 3:

Input:

s = "1010101010"

Output:

25

Explanation:

Only substrings that end with digit

'1'

are divisible by their last digit. There are 25 such substrings.

Constraints:

1 <= s.length <= 10

5

s

consists of digits only.

Code Snippets

C++:

```
class Solution {
public:
long long countSubstrings(string s) {

}
};
```

Java:

```
class Solution {
public long countSubstrings(String s) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public long CountSubstrings(string s) {

}
}
```

C:

```
long long countSubstrings(char* s) {

}
```

Go:

```
func countSubstrings(s string) int64 {
}
```

Kotlin:

```
class Solution {
fun countSubstrings(s: String): Long {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn count_substrings(s: String) -> i64 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def countSubstrings(s: String): Long = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Count Substrings Divisible By Last Digit
* Difficulty: Hard
* Tags: string, tree, 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:
long long countSubstrings(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Count Substrings Divisible By Last Digit
* Difficulty: Hard
* Tags: string, tree, 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 long countSubstrings(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Substrings Divisible By Last Digit
Difficulty: Hard
Tags: string, tree, 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 countSubstrings(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Substrings Divisible By Last Digit
* Difficulty: Hard
* Tags: string, tree, 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} s
* @return {number}
*/
var countSubstrings = function(s) {
};
```

TypeScript Solution:

```
/**
* Problem: Count Substrings Divisible By Last Digit
* Difficulty: Hard
* Tags: string, tree, 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 countSubstrings(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Count Substrings Divisible By Last Digit
* Difficulty: Hard
* Tags: string, tree, 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 long CountSubstrings(string s) {

}
}
```

C Solution:

```
/*
* Problem: Count Substrings Divisible By Last Digit
* Difficulty: Hard
* Tags: string, tree, 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
*/

long long countSubstrings(char* s) {

}
```

Go Solution:

```
// Problem: Count Substrings Divisible By Last Digit
// Difficulty: Hard
// Tags: string, tree, 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 countSubstrings(s string) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun countSubstrings(s: String): Long {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Count Substrings Divisible By Last Digit
// Difficulty: Hard
// Tags: string, tree, 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 count_substrings(s: String) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int countSubstrings(String s) {
}
}
```

Scala Solution:

```
object Solution {
def countSubstrings(s: String): Long = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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