Problem 2827: Number of Beautiful Integers in the Range

Problem Information

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

Problem Description

You are given positive integers

low

,

high

, and

k

.

A number is

beautiful

if it meets both of the following conditions:

The count of even digits in the number is equal to the count of odd digits.

The number is divisible by

k

.

Return

the number of beautiful integers in the range

[low, high]

.

Example 1:

Input:

low = 10, high = 20, k = 3

Output:

2

Explanation:

There are 2 beautiful integers in the given range: [12,18]. - 12 is beautiful because it contains 1 odd digit and 1 even digit, and is divisible by k = 3. - 18 is beautiful because it contains 1 odd digit and 1 even digit, and is divisible by k = 3. Additionally we can see that: - 16 is not beautiful because it is not divisible by k = 3. - 15 is not beautiful because it does not contain equal counts even and odd digits. It can be shown that there are only 2 beautiful integers in the given range.

Example 2:

Input:

low = 1, high = 10, k = 1

Output:

1

Explanation:

There is 1 beautiful integer in the given range: [10]. - 10 is beautiful because it contains 1 odd digit and 1 even digit, and is divisible by k = 1. It can be shown that there is only 1 beautiful integer in the given range.

Example 3:

Input:

low = 5, high = 5, k = 2

Output:

0

Explanation:

There are 0 beautiful integers in the given range. - 5 is not beautiful because it is not divisible by k = 2 and it does not contain equal even and odd digits.

Constraints:

0 < low <= high <= 10

9

0 < k <= 20

Code Snippets

C++:

```
class Solution {
public:
int numberOfBeautifulIntegers(int low, int high, int k) {

}
};
```

Java:

```
class Solution {
public int numberOfBeautifulIntegers(int low, int high, int k) {

}
}
```

Python3:

```
class Solution:
def numberOfBeautifulIntegers(self, low: int, high: int, k: int) -> int:
```

Python:

```
class Solution(object):
def numberOfBeautifulIntegers(self, low, high, k):
"""
:type low: int
:type high: int
:type k: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} low
* @param {number} high
* @param {number} k
* @return {number}
*/
var numberOfBeautifulIntegers = function(low, high, k) {

};
```

TypeScript:

```
function numberOfBeautifulIntegers(low: number, high: number, k: number):
number {

};
```

C#:

```
public class Solution {
public int NumberOfBeautifulIntegers(int low, int high, int k) {

}
}
```

C:

```
int numberOfBeautifulIntegers(int low, int high, int k) {

}
```

Go:

```
func numberOfBeautifulIntegers(low int, high int, k int) int {

}
```

Kotlin:

```
class Solution {
fun numberOfBeautifulIntegers(low: Int, high: Int, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func numberOfBeautifulIntegers(_ low: Int, _ high: Int, _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_beautiful_integers(low: i32, high: i32, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} low
# @param {Integer} high
# @param {Integer} k
# @return {Integer}
def number_of_beautiful_integers(low, high, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer $low
* @param Integer $high
* @param Integer $k
* @return Integer
*/
function numberOfBeautifulIntegers($low, $high, $k) {

}
}
```

Dart:

```
class Solution {
int numberOfBeautifulIntegers(int low, int high, int k) {

}
}
```

Scala:

```
object Solution {
def numberOfBeautifulIntegers(low: Int, high: Int, k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_beautiful_integers(low :: integer, high :: integer, k ::
integer) :: integer
def number_of_beautiful_integers(low, high, k) do

end
end
```

Erlang:

```
-spec number_of_beautiful_integers(Low :: integer(), High :: integer(), K ::
integer()) -> integer().
number_of_beautiful_integers(Low, High, K) ->
.
```

Racket:

```
(define/contract (number-of-beautiful-integers low high k)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Beautiful Integers in the Range
* Difficulty: Hard
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int numberOfBeautifulIntegers(int low, int high, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Beautiful Integers in the Range
* Difficulty: Hard
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int numberOfBeautifulIntegers(int low, int high, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Beautiful Integers in the Range
Difficulty: Hard
Tags: dp, math

Approach: Dynamic programming with memoization or tabulation
Time Complexity: O(n * m) where n and m are problem dimensions
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def numberOfBeautifulIntegers(self, low: int, high: int, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numberOfBeautifulIntegers(self, low, high, k):
"""
:type low: int
:type high: int
```

```
:type k: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Beautiful Integers in the Range
* Difficulty: Hard
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number} low
* @param {number} high
* @param {number} k
* @return {number}
*/
var numberOfBeautifulIntegers = function(low, high, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Beautiful Integers in the Range
* Difficulty: Hard
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

function numberOfBeautifulIntegers(low: number, high: number, k: number):
number {
```

```
};
```

C# Solution:

```
/*
* Problem: Number of Beautiful Integers in the Range
* Difficulty: Hard
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int NumberOfBeautifulIntegers(int low, int high, int k) {

}
}
```

C Solution:

```
/*
* Problem: Number of Beautiful Integers in the Range
* Difficulty: Hard
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

int numberOfBeautifulIntegers(int low, int high, int k) {

}
```

Go Solution:

```
// Problem: Number of Beautiful Integers in the Range
// Difficulty: Hard
// Tags: dp, math
```

```
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

func numberOfBeautifulIntegers(low int, high int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfBeautifulIntegers(low: Int, high: Int, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfBeautifulIntegers(_ low: Int, _ high: Int, _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Beautiful Integers in the Range
// Difficulty: Hard
// Tags: dp, math
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn number_of_beautiful_integers(low: i32, high: i32, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} low
# @param {Integer} high
# @param {Integer} k
# @return {Integer}
def number_of_beautiful_integers(low, high, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $low
* @param Integer $high
* @param Integer $k
* @return Integer
*/
function numberOfBeautifulIntegers($low, $high, $k) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfBeautifulIntegers(int low, int high, int k) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfBeautifulIntegers(low: Int, high: Int, k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_beautiful_integers(low :: integer, high :: integer, k ::
integer) :: integer
def number_of_beautiful_integers(low, high, k) do

end
end
```

Erlang Solution:

```
-spec number_of_beautiful_integers(Low :: integer(), High :: integer(), K ::
integer()) -> integer().
number_of_beautiful_integers(Low, High, K) ->
.
```

Racket Solution:

```
(define/contract (number-of-beautiful-integers low high k)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```