Problem 1067: Digit Count in Range

Problem Information

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

Problem Description

Given a single-digit integer

d

and two integers

low

and

high

, return

the number of times that

d

occurs as a digit in all integers in the inclusive range

[low, high]

.

Example 1:

Input:

d = 1, low = 1, high = 13

Output:

6

Explanation:

The digit d = 1 occurs 6 times in 1, 10, 11, 12, 13. Note that the digit d = 1 occurs twice in the number 11.

Example 2:

Input:

d = 3, low = 100, high = 250

Output:

35

Explanation:

The digit d = 3 occurs 35 times in 103,113,123,130,131,...,238,239,243.

Constraints:

0 <= d <= 9

1 <= low <= high <= 2 * 10

8

Code Snippets

C++:

```
class Solution {
public:
int digitsCount(int d, int low, int high) {

}
};
```

Java:

```
class Solution {
public int digitsCount(int d, int low, int high) {

}
}
```

Python3:

```
class Solution:
def digitsCount(self, d: int, low: int, high: int) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function digitsCount(d: number, low: number, high: number): number {

};
```

C#:

```
public class Solution {
public int DigitsCount(int d, int low, int high) {

}
}
```

C:

```
int digitsCount(int d, int low, int high) {

}
```

Go:

```
func digitsCount(d int, low int, high int) int {

}
```

Kotlin:

```
class Solution {
fun digitsCount(d: Int, low: Int, high: Int): Int {

}
}
```

Swift:

```
class Solution {
func digitsCount(_ d: Int, _ low: Int, _ high: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn digits_count(d: i32, low: i32, high: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} d
# @param {Integer} low
# @param {Integer} high
# @return {Integer}
def digits_count(d, low, high)

end
```

PHP:

```
class Solution {

/**
* @param Integer $d
* @param Integer $low
* @param Integer $high
* @return Integer
*/
function digitsCount($d, $low, $high) {

}
}
```

Dart:

```
class Solution {
int digitsCount(int d, int low, int high) {

}
}
```

Scala:

```
object Solution {
def digitsCount(d: Int, low: Int, high: Int): Int = {
```

```
}
}
```

Elixir:

```
defmodule Solution do
@spec digits_count(d :: integer, low :: integer, high :: integer) :: integer
def digits_count(d, low, high) do

end
end
```

Erlang:

```
-spec digits_count(D :: integer(), Low :: integer(), High :: integer()) ->
integer().
digits_count(D, Low, High) ->
.
```

Racket:

```
(define/contract (digits-count d low high)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Digit Count in 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 digitsCount(int d, int low, int high) {

}
};
```

Java Solution:

```
/**
* Problem: Digit Count in 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 digitsCount(int d, int low, int high) {

}
}
```

Python3 Solution:

```
"""
Problem: Digit Count in 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 digitsCount(self, d: int, low: int, high: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Digit Count in 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} d
* @param {number} low
* @param {number} high
* @return {number}
*/
var digitsCount = function(d, low, high) {

};
```

TypeScript Solution:

```
/**
* Problem: Digit Count in 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 digitsCount(d: number, low: number, high: number): number {

};
```

C# Solution:

```
/*
* Problem: Digit Count in 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 DigitsCount(int d, int low, int high) {

}
}
```

C Solution:

```
/*
* Problem: Digit Count in 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 digitsCount(int d, int low, int high) {

}
```

Go Solution:

```
// Problem: Digit Count in 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 digitsCount(d int, low int, high int) int {

}
```

Kotlin Solution:

```
class Solution {
fun digitsCount(d: Int, low: Int, high: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func digitsCount(_ d: Int, _ low: Int, _ high: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Digit Count in 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 digits_count(d: i32, low: i32, high: i32) -> i32 {
```

```
}
}
```

Ruby Solution:

```
# @param {Integer} d
# @param {Integer} low
# @param {Integer} high
# @return {Integer}
def digits_count(d, low, high)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $d
* @param Integer $low
* @param Integer $high
* @return Integer
*/
function digitsCount($d, $low, $high) {

}
}
```

Dart Solution:

```
class Solution {
int digitsCount(int d, int low, int high) {

}
}
```

Scala Solution:

```
object Solution {
def digitsCount(d: Int, low: Int, high: Int): Int = {
```

```
}
}
```

Elixir Solution:

```
defmodule Solution do
@spec digits_count(d :: integer, low :: integer, high :: integer) :: integer
def digits_count(d, low, high) do

end
end
```

Erlang Solution:

```
-spec digits_count(D :: integer(), Low :: integer(), High :: integer()) ->
integer().
digits_count(D, Low, High) ->
.
```

Racket Solution:

```
(define/contract (digits-count d low high)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```