Problem 357: Count Numbers with Unique Digits

Problem Information

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

Problem Description

Given an integer

n

, return the count of all numbers with unique digits,

x

, where

0 <= x < 10

n

.

Example 1:

Input:

n = 2

Output:

91

Explanation:

The answer should be the total numbers in the range of 0 ≤ x < 100, excluding 11,22,33,44,55,66,77,88,99

Example 2:

Input:

n = 0

Output:

1

Constraints:

0 <= n <= 8

Code Snippets

C++:

```
class Solution {
public:
int countNumbersWithUniqueDigits(int n) {

}
};
```

Java:

```
class Solution {
public int countNumbersWithUniqueDigits(int n) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public int CountNumbersWithUniqueDigits(int n) {

}
}
```

C:

```
int countNumbersWithUniqueDigits(int n) {

}
```

Go:

```
func countNumbersWithUniqueDigits(n int) int {

}
```

Kotlin:

```
class Solution {
fun countNumbersWithUniqueDigits(n: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def countNumbersWithUniqueDigits(n: Int): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (count-numbers-with-unique-digits n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Numbers with Unique Digits
* Difficulty: Medium
* 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 countNumbersWithUniqueDigits(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Count Numbers with Unique Digits
* Difficulty: Medium
* 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 countNumbersWithUniqueDigits(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Numbers with Unique Digits
Difficulty: Medium
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 countNumbersWithUniqueDigits(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Numbers with Unique Digits
* Difficulty: Medium
* 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} n
* @return {number}
*/
var countNumbersWithUniqueDigits = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Numbers with Unique Digits
* Difficulty: Medium
* 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 countNumbersWithUniqueDigits(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Count Numbers with Unique Digits
* Difficulty: Medium
* 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 CountNumbersWithUniqueDigits(int n) {

}
}
```

C Solution:

```
/*
* Problem: Count Numbers with Unique Digits
* Difficulty: Medium
* 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 countNumbersWithUniqueDigits(int n) {

}
```

Go Solution:

```
// Problem: Count Numbers with Unique Digits
// Difficulty: Medium
// 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 countNumbersWithUniqueDigits(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countNumbersWithUniqueDigits(n: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Count Numbers with Unique Digits
// Difficulty: Medium
// 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 count_numbers_with_unique_digits(n: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

```
object Solution {
def countNumbersWithUniqueDigits(n: Int): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (count-numbers-with-unique-digits n)
(-> exact-integer? exact-integer?)
)
```