Problem 2376: Count Special Integers

Problem Information

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

Problem Description

We call a positive integer

special

if all of its digits are

distinct

.

Given a

positive

integer

n

, return

the number of special integers that belong to the interval

[1, n]

.

Example 1:

Input:

n = 20

Output:

19

Explanation:

All the integers from 1 to 20, except 11, are special. Thus, there are 19 special integers.

Example 2:

Input:

n = 5

Output:

5

Explanation:

All the integers from 1 to 5 are special.

Example 3:

Input:

n = 135

Output:

110

Explanation:

There are 110 integers from 1 to 135 that are special. Some of the integers that are not special are: 22, 114, and 131.

Constraints:

1 <= n <= 2 * 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

```
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int countSpecialNumbers(int n) {

}
```

Go:

```
func countSpecialNumbers(n int) int {

}
```

Kotlin:

```
class Solution {
fun countSpecialNumbers(n: Int): Int {
}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (count-special-numbers n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Special Integers
* 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 countSpecialNumbers(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Count Special Integers
* 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 countSpecialNumbers(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Special Integers
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 countSpecialNumbers(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Special Integers
* 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} n
* @return {number}
*/
var countSpecialNumbers = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Special Integers
* 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 countSpecialNumbers(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Count Special Integers
* 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 CountSpecialNumbers(int n) {

}
}
```

C Solution:

```
/*
* Problem: Count Special Integers
* 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 countSpecialNumbers(int n) {

}
```

Go Solution:

```
// Problem: Count Special Integers
// 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 countSpecialNumbers(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Count Special Integers
// 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 count_special_numbers(n: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (count-special-numbers n)
(-> exact-integer? exact-integer?)
)
```