Problem 2048: Next Greater Numerically Balanced Number

Problem Information

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

Problem Description

An integer

x

is

numerically balanced

if for every digit

d

in the number

x

, there are

exactly

d

occurrences of that digit in

x

.

Given an integer

n

, return

the

smallest numerically balanced

number

strictly greater

than

n

.

Example 1:

Input:

n = 1

Output:

22

Explanation:

22 is numerically balanced since: - The digit 2 occurs 2 times. It is also the smallest numerically balanced number strictly greater than 1.

Example 2:

Input:

n = 1000

Output:

1333

Explanation:

1333 is numerically balanced since: - The digit 1 occurs 1 time. - The digit 3 occurs 3 times. It is also the smallest numerically balanced number strictly greater than 1000. Note that 1022 cannot be the answer because 0 appeared more than 0 times.

Example 3:

Input:

n = 3000

Output:

3133

Explanation:

3133 is numerically balanced since: - The digit 1 occurs 1 time. - The digit 3 occurs 3 times. It is also the smallest numerically balanced number strictly greater than 3000.

Constraints:

0 <= n <= 10

6

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int nextBeautifulNumber(int n) {

}
```

Go:

```
func nextBeautifulNumber(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (next-beautiful-number n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Next Greater Numerically Balanced Number
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int nextBeautifulNumber(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Next Greater Numerically Balanced Number
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

class Solution {
public int nextBeautifulNumber(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Next Greater Numerically Balanced Number
Difficulty: Medium
Tags: math, hash

Approach: Use hash map for O(1) lookups
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(n) for hash map
"""

class Solution:
def nextBeautifulNumber(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Next Greater Numerically Balanced Number
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Next Greater Numerically Balanced Number
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

function nextBeautifulNumber(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Next Greater Numerically Balanced Number
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int NextBeautifulNumber(int n) {

}
}
```

C Solution:

```
/*
* Problem: Next Greater Numerically Balanced Number
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

int nextBeautifulNumber(int n) {

}
```

Go Solution:

```
// Problem: Next Greater Numerically Balanced Number
// Difficulty: Medium
// Tags: math, hash
//
// Approach: Use hash map for O(1) lookups
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(n) for hash map
```

```
func nextBeautifulNumber(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Next Greater Numerically Balanced Number
// Difficulty: Medium
// Tags: math, hash
//
// Approach: Use hash map for O(1) lookups
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (next-beautiful-number n)
(-> exact-integer? exact-integer?)
)
```