



# **CSA0695 Design and Analysis of Algorithms for Open Addressing Techniques**

## **capstone project**

### **Next Greater Numerically Balanced Number**


**Presented by: S.Ajay kumar reddy(192211480)**  
**guided by: Dr. R. Dhanalakshmi**

# Next Greater Numerically Balanced Number

## PROBLEM STATEMENT:

- Next Greater Numerically Balanced Number 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.

# ABSTRACT:

- This capstone project aims to develop an algorithm that finds the smallest numerically balanced number greater than a given integer  $n$ .
  - A numerically balanced number is defined as one in which each digit  $d$  appears exactly  $d$  times within the number. For example, the number 22 is numerically balanced because the digit 2 occurs exactly 2 times.
  - The project will focus on creating an efficient solution that can quickly identify such numbers, even for large inputs, by analyzing digit frequency and leveraging mathematical properties. This work has potential applications in number theory, pattern recognition, and combinatorial analysis.
- 

## INTRODUCTION:

Numerically balanced numbers are an intriguing class of integers where the count of each digit in the number corresponds exactly to the digit's value.

For instance, in the number 22

- The digit 2 appears two times, satisfying the numerically balanced condition. While these numbers follow a straightforward definition, finding such numbers—especially those greater than a given integer—requires an in-depth exploration of digit patterns and frequency analysis.
- This problem introduces complexities related to number theory and algorithmic design, making it a compelling topic for computational mathematics.
- The ability to split a string into valid numerical sequences with specific properties requires both a strong understanding of string manipulation and recursive problem-solving techniques.

# OUTPUT:

- The provided C code is designed to find the smallest numerically balanced number that is strictly greater than a given integer n. The code utilizes a combination of digit frequency analysis and iterative checking to solve this problem. The countDigits function counts the occurrences of each digit in the number, while the isNumericallyBalanced function verifies whether a number meets the numerically balanced criteria. The nextBalancedNumber function increments the given integer and uses these functions to efficiently identify the next numerically balanced number

```
Enter a number: 6
The next numerically balanced number greater than 6 is 22
-----
Process exited after 2.988 seconds with return value 0
Press any key to continue . . . |
```

## COMPLEXITY ANALYSIS:

**Time Complexity:** The time complexity of finding the next numerically balanced number is primarily influenced by the number of increments required to find the next valid number and the cost of checking each number. Each check involves counting digits, which has a time complexity of  $O(d)$ , where  $d$  is the number of digits.

### BEST CASE:

In the best-case scenario, the next numerically balanced number is found immediately after  $n$ , leading to a complexity of  $O(d)$ , where  $d$  is the number of digits in  $n$ . This occurs when the next number after  $n$  is numerically balanced without requiring many increments

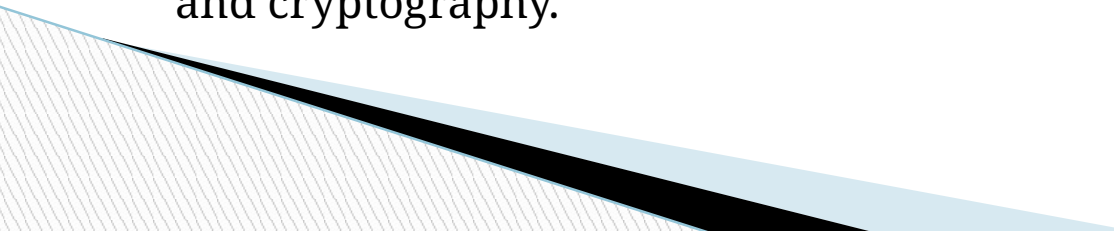
### WORST CASE:

The worst-case scenario involves a large number of increments before finding the next balanced number. Here, the time complexity can be approximated as  $O(k * d)$ , where  $k$  represents the number of increments needed and  $d$  is the number of digits in the current number.

## **AVERAGE CASE:**

- On average, the time complexity depends on the typical gap between consecutive numerically balanced numbers. If this gap is reasonably small, the complexity remains closer to  $O(d)$ . However, this average can vary based on the distribution of numerically balanced numbers.

## **FUTURE SCOPE:**

- Future work could focus on optimizing the search for numerically balanced numbers, possibly through heuristic or mathematical methods to predict gaps between balanced numbers. Additionally, expanding the concept to other types of digit-based constraints could provide further insights into combinatorial number theory and practical applications in data analysis and cryptography.
- 

## CONCLUSION:

- The project provides a practical approach to finding the next numerically balanced number, offering insights into digit-based number properties and computational methods. While the algorithm is efficient in terms of space, its time complexity varies based on the number of increments needed to find the next balanced number. Future improvements could enhance performance and explore broader applications of numerically balanced numbers in computational fields.
- 