Final Project Report

Group 2: Khaing Thazin Phyoe and Min Thant Hein

Parami University

DATA 301: Data Structures and Algorithms

Professor Aye Hnin Khine

May 24, 2025

Final Project Report

Problem Statement

The Palindromic Number Guessing Game aims to provide users with an engaging and unique twist on the traditional number guessing game. In this game, users are challenged to guess a randomly generated palindromic number within a specified range. A palindromic number reads the same backward as forward, such as 121 or 1331, adding an extra layer of challenge and intrigue to the gameplay. The primary objective of this project is to develop a Python program that not only generates a random palindromic number but also efficiently manages and manipulates data to enhance the gaming experience. The game should provide users with a limited number of attempts to guess the correct number, making the challenge more exciting.

Data Structures Used

The program uses several data structures to manage and manipulate data efficiently. One of the primary data structures used is a binary search tree. BST allows the efficient search and storage of guesses to give hints. Additionally, a Queue is utilized to provide hints based on the first digit of the player's guesses. Each guess is added to the Queue, and the game checks the first digit of these guesses to offer hints. For instance, if the first digit of a guess matches the first digit of the secret number, the game provides a hint indicating that the number starts with that digit. This use of the Queue ensures that hints are provided based on the most recent guesses, making the game more dynamic and user-friendly. By integrating both the BST and the Queue, the program efficiently manages data and offers helpful hints, creating a more engaging and interactive gaming experience. Moreover, Linkedlist data structure is used because it allows storing the sequence of guesses. Thus, Linkedlist data structure is used in implementing GuessHistory class. Additionally, Stack data structure is applied inside the Stack class. This is because the Last-in-First-Out principle is a good fit for the ‘undo’ feature. Specifically, the ‘undo’ feature reverses the most recent action.

Overall Time Complexity

Data structure operations like adding to the linked list history and showing the history have linear time complexity (O(n)) since the time complexity grows alongside the input, the space complexity. Stack operations, such as push, and pop, are generally constant time O(1), since it takes constant time to execute.

Queue Operations:

Enqueue: O(1) average, O(n) worst case.
Dequeue: O(n).
Peek: O(1).
Size: O(1).

BST Operations:

Insert: O(h), where h is the height of the tree.
In-Order Traversal: O(n).

Conclusion

The Number Guessing Game is a well-structured and engaging program that uses various data structures and algorithms to provide an efficient and enjoyable gaming experience. The use of binary search in the reverse guessing game ensures optimal performance, and the modular design of the code makes it easy to maintain and extend. The program also includes features to track and display user statistics, adding an element of competition and motivation for the user. Overall, the Number Guessing Game is a great example of how data structures and algorithms can be used to create an engaging and efficient program.