Reflection on Assignment 1: Data Structures Adventure

Name: Rupak Poudyal

Student ID: C0903770

GitHub Link: https://github.com/rupakpoudyal5/DataStructures-Assignment1-Rupak_Poudyal

Challenge 1: The Array Artifact

What I Learned: In this challenge, I reinforced my understanding of arrays, especially when it comes to managing fixed-size collections and using different search algorithms (linear search and binary search). It also helped me understand the importance of sorting when performing binary search, as it significantly improves the efficiency of searching compared to linear search.

Difficulties Encountered: One difficulty I encountered was handling the binary search since it requires the array to be sorted. Initially, I overlooked this requirement and was confused by unexpected results. After realizing this, I included sorting in the method to ensure binary search works correctly.

Ideas for Improvement: To extend the solution, I could implement dynamic resizing of the array to allow for a more flexible artifact vault that can grow when full. Another potential improvement is to allow the vault to store multiple types of objects (using generics), not just strings.

Challenge 2: The Linked List Labyrinth

What I Learned: Through this challenge, I learned how to implement a basic singly linked list and work with dynamic memory allocation for nodes. The loop detection part taught me about Floyd's Cycle Detection algorithm, a useful technique for detecting cycles in linked structures.

Difficulties Encountered: The challenge in this task was managing the deletion of the last node. Initially, I made a mistake by not properly handling the edge case when the list had only one element. I resolved this by carefully considering both the single-node case and the general case.

Ideas for Improvement: I could further improve the labyrinth by adding a method to reverse the path, enabling us to backtrack through the labyrinth. Another possible extension is to allow for doubly linked lists, which would make it easier to traverse backward through the path.

Challenge 3: The Stack of Ancient Texts

What I Learned: This challenge reinforced my understanding of the stack data structure and its last-in, first-out (LIFO) behavior. Implementing common stack operations like push, pop, and peek helped me see how a stack can be useful for managing temporary data.

Difficulties Encountered: A minor issue I encountered was ensuring that popping from an empty stack didn't cause an error. I implemented a check to ensure the stack was not empty before allowing pop operations, which fixed the issue.

Ideas for Improvement: To extend this solution, I could implement a more advanced stack with a maximum size limit. This would add complexity by requiring overflow handling. Additionally, I could create a feature that logs the history of popped items for future reference.

Challenge 4: The Queue of Explorers

What I Learned: In this challenge, I learned about the circular queue and its implementation using a fixed-size array. It was interesting to see how the circular nature of the queue allows efficient use of space, as it wraps around when it reaches the end of the array.

Difficulties Encountered: One difficulty I faced was understanding how to handle the wrap-around behavior when the rear of the queue reaches the end of the array. After some debugging, I was able to correctly manage the circular nature of the queue by using the modulo operator to reset the index.

Ideas for Improvement: One improvement could be the addition of dynamic resizing for the queue, allowing the queue to expand as needed. Another idea is to implement a priority queue, where explorers with higher priority (e.g., based on experience) are dequeued first.

Challenge 5: The Binary Tree of Clues

What I Learned: Implementing the binary tree helped me understand how recursive methods work for both insertion and traversal (in-order, pre-order, post-order). The challenge also provided insight into the utility of binary search trees for organizing and searching data efficiently.

Difficulties Encountered: Initially, I struggled with implementing the recursive traversal methods, particularly pre-order and post-order. After reviewing the recursive logic carefully, I was able to implement them correctly. The key was ensuring that the order of recursive calls was correct.

Ideas for Improvement: To improve the binary tree, I could implement a balancing mechanism (e.g., AVL tree or Red-Black tree) to ensure the tree remains balanced and operations are performed efficiently. Another extension would be to implement breadth-first traversal and visualize the tree structure.

Conclusion

This assignment was a great opportunity to deepen my understanding of fundamental data structures. Each challenge highlighted different aspects of these structures and their practical applications. I also improved my problem-solving skills as I worked through the challenges and overcame various difficulties. In the future, I hope to extend these solutions by making them more dynamic and flexible for real-world applications.