Reflection on Data Structures Assignment 1

Name: Jashandeep Kaur
Student ID: C0913241
Course: Data Structures

Learnings from Each Challenge

1. ArtifactVault

In implementing the ArtifactVault class, I gained a deeper understanding of array data structures and their limitations. Managing a fixed-size array taught me about the importance of capacity and memory management. I learned how to implement both linear and binary search algorithms, which enhanced my problem-solving skills and my ability to optimize search operations.

2. LabyrinthPath

The LabyrinthPath class allowed me to explore linked lists. I appreciated how linked lists can grow and shrink dynamically, providing flexibility that arrays lack. Implementing the loop detection algorithm was particularly enlightening, as it deepened my understanding of algorithmic efficiency and memory utilization.

3. ScrollStack

Creating the ScrollStack class provided practical experience with stack data structures. I learned about Last In First Out (LIFO) principles and the importance of stack operations like push and pop. This experience was crucial for understanding how stacks can be applied in various programming scenarios, such as managing function calls and undo mechanisms.

4. ExplorerQueue

The implementation of the ExplorerQueue class introduced me to circular queues. I learned how to manage the front and rear pointers effectively, ensuring efficient use of space. This exercise reinforced my understanding of queue operations and the importance of boundary conditions when implementing circular data structures.

5. ClueTree

Building the ClueTree class was a comprehensive exercise in tree data structures. I learned how to implement binary search trees (BSTs) and the significance of traversal methods (in-order, pre-order, post-order). This experience highlighted the efficiency of trees for searching and storing hierarchical data.

Difficulties Encountered

One of the significant challenges I faced was debugging the binary search implementation in the ArtifactVault class. Initially, I struggled with ensuring the array was sorted before performing the search, which led to incorrect results. After revisiting the sorting logic and thoroughly testing the search methods, I was able to resolve the issue. This experience taught me the importance of rigorous testing and validation of algorithms.

Another challenge was managing the linked list in the LabyrinthPath class, particularly when implementing the removal of nodes. I had to ensure that pointers were correctly updated to avoid memory leaks. This reinforced the importance of careful memory management when dealing with dynamic data structures.

Ideas for Improvement

Reflecting on my implementations, I see several areas for potential improvement:

1. **ArtifactVault**: I could enhance the ArtifactVault class by implementing dynamic resizing for the array, allowing it to expand as needed instead of having a fixed size. This would provide more flexibility in managing the artifacts.
2. **LabyrinthPath**: I could improve the LabyrinthPath class by adding methods for reversing the path or finding the shortest path using algorithms like Dijkstra's or A*.
3. **ScrollStack**: Adding an additional method to count the number of scrolls in the stack could be beneficial for tracking purposes.
4. **ExplorerQueue**: I could implement error handling for when the queue is empty or full to provide better user feedback and avoid exceptions.
5. **ClueTree**: Implementing balancing techniques for the ClueTree, such as AVL or Red-Black trees, could enhance search efficiency, especially with larger datasets.

Conclusion

This assignment provided valuable insights into various data structures and their practical implementations. Each challenge contributed to my growth as a programmer, enhancing my understanding of algorithms and data management techniques. I am eager to apply these skills to future projects and continue exploring advanced data structures and algorithms.