CS 3303-01 Data Structures

Instructor: Professor Vandelun Amokaha

Name: Ryohei Hayashi

Learning Journal 5

**Summary of the Week**

This week, I studied the general tree structures and their terminology, becoming familiar with the definitions and characteristics of trees, subtrees, and forests. I focused especially on being able to clearly explain the differences between binary trees and general trees, while also learning how to implement trees using both array and linked list structures. Furthermore, I deepened my understanding of various tree traversal methods, the parent pointer approach, and the implementation of special trees such as K-ary trees and sequential trees.

**Personal Reflections**

This week’s learning involved repeatedly studying the basic theories and implementation of tree structures. It required me not only to understand the theory but also to think about how to apply it to actual algorithm design and coding. While reviewing fundamental concepts such as trees and subtrees, I had the opportunity to delve into more specialized tree structures, like K-ary trees and sequential trees, and consider the efficiency and use cases of each data structure. Additionally, I enhanced my understanding by not only using Jeliot to visualize the algorithm's behavior but also by manually checking input and output using an online Java compiler. This helped deepen my comprehension.

The most challenging aspect was encountering unfamiliar tree structures like K-ary trees and sequential trees for the first time. Unlike the standard binary tree, it took time to grasp their specific use cases and the advantages and disadvantages of their implementation. However, I also felt a sense of accomplishment as my understanding deepened with the learning process. Drawing diagrams to visually understand tree structures and checking the program's operation were effective steps in solidifying my understanding.

**Topics Studied in Depth**

This week, I focused particularly on the theory and implementation of tree structures. After reviewing basic terms like trees, subtrees, and forests, I proceeded to implement these structures using arrays and linked lists. One particularly interesting aspect was the implementation of tree structures using the parent pointer approach. This method, which involves referencing the parent node, proved to be an efficient way to manage parent-child relationships and offered significant advantages in terms of memory efficiency and computational cost. Additionally, studying the implementation of special trees like K-ary trees and sequential trees provided me with a good opportunity to understand how these structures are suited for specific use cases and datasets.

In the case of K-ary trees, I learned that allowing each node to have up to K children can help create a balanced tree, making traversal algorithms and insertion/deletion operations more efficient. On the other hand, I realized that as K increases, memory usage also rises, presenting a challenge in selecting an appropriate value for K during implementation. Regarding sequential trees, I found it particularly striking that the order of the data allows for efficient data management, and the efficiency of the tree depends largely on the insertion order and traversal methods.

Through this study, I gained a renewed understanding that the choice of data structure directly impacts algorithm performance. By connecting theory with practice, I am gradually developing the ability to apply textbook knowledge to real-world problems.

**Future Challenges and Goals for Next Week**

Next week, I plan to focus on learning sorting algorithms. I will first aim to understand the mechanisms of basic sorting algorithms such as insertion sort, bubble sort, and selection sort, and be able to clearly explain their inefficiencies. Furthermore, I plan to implement advanced algorithms like shell sort and quicksort while evaluating their efficiency through asymptotic analysis and understanding the cost in terms of processing time. By mastering these concepts, I hope to acquire the skills to handle more complex data structures and algorithms effectively.

Word Count: 606

References

1. Shaffer, C. (2011). *A Practical Introduction to Data Structures and Algorithm Analysis.* Blacksburg: Virginia. Tech.