CS 3303-01 Data Structures

Instructor: Professor Vandelun Amokaha

Name: Ryohei Hayashi

Learning Journal 2

**Summary of the Week**

This week, I studied the basic concepts of asymptotic analysis of algorithms. In particular, I learned how Big O, Big Omega, and Big Theta notations are used to evaluate the performance of algorithms. I also learned about the simplification rules described in section 3.4.4 of the Shaffer text, which provided valuable knowledge for analyzing complex algorithms. Furthermore, I reaffirmed the importance of considering the trade-offs between cost and benefit in algorithm design, and studied how to determine the performance of algorithms in best, worst, and average cases. Through this, I was able to deepen my practical understanding.

**Personal Reflections**

Through this week’s learning, I was able to gain a deeper understanding of the concept of asymptotic analysis in evaluating algorithms. In particular, I reaffirmed the need to carefully organize the input data and conditions to be aware of the trade-offs between cost and benefit when designing algorithms. In fields such as web services, these elements change depending on the growth and scale of the service, and I found it fascinating that improvement in design and algorithms is constantly required. As a hobbyist in competitive programming, I had a certain level of understanding of Big O notation, but I realized I hadn’t fully grasped Big Omega and Big Theta notations. This week’s learning allowed me to gain new perspectives on these notations, making it a highly valuable experience.

**Topics Studied in Depth**

This week, I delved deeper into Big O, Big Omega, and Big Theta notation, which play a crucial role in algorithm evaluation, by using specific functions as examples.

First, Big O notation represents the upper bound of an algorithm’s time or space complexity in the worst-case scenario. For example, in the function , the dominant term is , and thus, the Big O of this function is. This means that as the input size increases, the execution time grows by a factor of in the worst case.

Next, Big Omega notation shows the lower bound of an algorithm in the best-case scenario. For the same function , the smallest time it takes corresponds to the linear term . Therefore, the Big Omega is , indicating that even in the best case, the time complexity will grow by at least .

Finally, Big Theta notation is used when the upper and lower bounds are the same. In this case, for , the dominant term is , so the Big Theta of this function is .This suggests that the algorithm’s performance grows consistently in both best and worst cases.

By using these specific functions, I was able to better understand how these notations are applied in algorithm evaluation. Particularly, I reaffirmed their importance in determining how efficiently an algorithm will perform across various scenarios when designing or choosing algorithms.

**Future Challenges and Goals for Next Week**

Next week, I will be studying list structures and linked list structures. List structures are foundational data structures that allow for continuous or dynamic storage of elements, such as arrays and linked lists. Linked lists, in particular, enable efficient dynamic memory allocation, insertion, and deletion of elements. This flexibility makes them ideal for managing data in various algorithms and applications. The goal for next week’s study is to gain a deeper understanding of the basic operations of these data structures and how they affect time and space efficiency. Additionally, I aim to apply these structures in future competitive programming to design more efficient algorithms.

Word Count: 593

References

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