Lessons Learned Reflection

Alejandro Ricciardi

Colorado State University Global

CSC400: Data Structures and Algorithms

Professor: Hubert Pensado

October 06, 2024

Lessons Learned Reflection

The Computer Science CSC400: Data Structures and Algorithms course took me through a learning journey that was challenging and very satisfying. It introduced me to concepts in data structures and algorithms data were novel to me. In this reflection, I will summarize key lessons that I learned and reflect on how these lessons can be applied to more effective coding as I continue my professional and academic endeavors.

Data Structures in Problem Solving

One of the most important concepts that I learned is the concept of Abstract Data Types (ADTs), such as bags, stacks, queues, deque, and priority queues. These data structures play a critical role in providing powerful frameworks and interfaces for organizing and managing data. At the beginning of the course, I was introduced to the importance of choosing the right data structure as it serves as the foundation, the cornerstone, upon which to build functional, efficient, and reliable applications. For example, using a stack (Last In, First Out) or a queue (First In, First Out) can drastically influence how an application functions and simplify problems that involve processing elements in a specific order, such as expression evaluation or task scheduling, to meet the goal of the application. On the other hand, picking the wrong data structure can lead to inefficient code, increased complexity, issues performance, and resource wasting, in the worst cases, completely failing to meet the application's goals.

Algorithm Efficiency and Big-O Notation

This part is what enjoyed the most about the course, but it was also the most challenging part. Algorithm efficiency is crucial for the overall performance of an application as it dictates how quickly a program can solve a problem and how many resources it consumes, especially as the size of the input grows. Thus, evaluating algorithms efficiently is crucial, CSC400 introduced

me to complexity analysis, which is a method to analyze and categorize algorithms into best, worst, and average cases using the Big-O notation. Big-O notation analyzes both time (completion time) and space complexity (memory resources). For example, the selection sort algorithm is not a very complex algorithm making it easy to implement, but it has a time complexity of $O(n^2)$, making it not suited to sort large datasets. On the other hand, more complex and advanced algorithms such as quick sort which has a time complexity of $O(n \log n)$ is better suited to sort large data sets, but it is more difficult to implement. Learning about the algorithm complexity allows me to analyze my code improving the efficiency of applications; it also improved my understanding of advanced algorithms such as quick sort and merge sort.

Recursion

CSC400 also introduced me to the concept of recursion. Recursion is the process of a function calling itself, which is particularly useful for solving problems that can be broken down into smaller subproblems, such as searching, sorting, and traversing tree structures. Recursion is very useful for crafting powerful algorithms; however, it has advantages such as the risk of stack overflow if the recursive depth (number of recursion calls) becomes too large. Nonetheless, recursion is a powerful tool, and in CSC400, I further my understanding of recursion and how and when it is appropriate to use it, making me a more efficient coder.

Sorting Algorithms

The class furthered my understanding and knowledge of sorting algorithms. I got the opportunity to explore the difference between various algorithms such as the difference between selection sort, insertion sort, shell sort, quick sort, and radix sort. I also had the opportunity to implement those algorithms. Additionally, I learned how to compare and evaluate their implementation into different applications based on their performance and factors such as

stability, time complexity, and suitability for various data types. Combined with my newly acquired knowledge of data structures, now, I can build more efficient, functional, and robust applications.

Beyond CSC400

I thoroughly enjoyed the course, I found data structures and algorithms fascinating. The knowledge and understanding that acquired in this class strengthened my skills as a coder and my foundations as a future software engineer by teaching me how to choose the appropriate data structures, optimize algorithms for efficiency using Big-O notation, and apply recursion for solving complex problems. In other words, CSC400 has equipped me with the skills and knowledge to take on more advanced programming challenges with confidence and a deeper understanding of how to write more efficient, functional, and robust code; which is essential for my future in Computer Science.