**Learned in Data Structures and Algorithms**

Agustin Aguayo

Colorado State University Global

CSC400: Data Structures and Algorithms

Angelo Luo

May 11th, 2025

**Lessons Learned in Data and Algorithms**

**This programming course has been a journey through increasingly complex topics in Java, data structures, and algorithms. Building on the Introduction to Programming II course, I expanded my skills from basic Java programming and abstract data types (ADTs) into more advanced topics, focusing on user-defined data structures, sorting algorithms, and recursive methods. The hands-on projects in my project folder allowed me to directly apply these lessons, deepening my understanding and improving my coding abilities.**

**Valuable Lessons Learned**

**In Module 1, I was introduced to more advanced uses of abstract data types, specifically the bag data type in Java. I learned how a bag allows for duplicate elements and how it differs from other collections like sets. This provided a strong foundation for understanding how data can be stored and manipulated in different ways. Applying this to my projects, I practiced creating custom bag-like data structures, which helped reinforce the key concepts of data encapsulation and generic programming as described by Lysecky and Vahid (2022).**

**Module 2 built upon this by exploring the different ways to implement bags, including fixed-size arrays, dynamic arrays, and linked lists. This module was key in helping me understand the trade-offs between these implementations, particularly in terms of memory use and flexibility. Implementing bags using linked lists in my projects gave me practical experience with node management and pointer logic, which is essential for writing efficient and scalable code.**

**In Module 3, I learned about complexity analysis and the use of Big O notation to evaluate algorithm performance. Goodrich, Tamassia, and Goldwasser (2022) explain that understanding algorithm efficiency is critical in selecting the right approach to solve a problem. This was especially useful when working on sorting and recursive methods later in the course. I applied Big O analysis in my project folder when evaluating different sorting methods and recursive solutions, helping me see the importance of performance in real-world coding.**

**Module 4 introduced the stack data structure and its typical operations such as push, pop, and peek. I implemented my own stack class, which gave me a clear understanding of how stacks manage data using a last-in, first-out approach. I also used the ADT Stack class in Java and wrote code that applied stack operations to solve problems. This experience strengthened my knowledge of how stacks can be used in both algorithm development and problem-solving situations.**

**In Module 5, I explored recursive algorithms and their implementations. Writing recursive methods helped me understand the importance of defining clear base cases and ensuring that recursion does not lead to infinite loops or stack overflow errors. I practiced calculating the time complexity of recursive algorithms, further building on my understanding of Big O notation from Module 3. I also learned that some problems, like traversing linked lists or implementing quick sort, are naturally suited to recursive approaches.**

**Module 6 focused on sorting algorithms, including selection sort, insertion sort, and shell sort. I implemented these algorithms in Java and evaluated their performance using small datasets. This hands-on approach allowed me to see how each algorithm works step by step and how they perform in different scenarios. The lessons from this module were directly applied in my project folder when I wrote programs that sorted data and analyzed the results.**

**Finally, in Module 7, I worked with advanced sorting algorithms such as merge sort, quick sort, and radix sort. For my final project, I specifically used quick sort to sort a queue of 'Person' objects by last name and age in descending order. Implementing quick sort helped me understand the divide-and-conquer strategy and how recursive partitioning works. Goodrich, Tamassia, and Goldwasser (2022) highlight the efficiency of quick sort in many scenarios, and my project showed me firsthand how powerful this algorithm can be when sorting custom objects.**

**Throughout the course, I also improved my coding practices by writing clean, well-documented code using comments and meaningful names. I applied 'try-catch' blocks for error handling to make my programs more user-friendly and reliable. I also used Git for version control, keeping my work organized and allowing me to track progress and changes effectively.**

**Conclusion**

**In conclusion, the lessons from each module combined with the projects in my folder gave me practical experience in advanced Java programming, data structures, algorithms, recursion, and performance analysis. By applying the concepts from Lysecky and Vahid (2022) and Goodrich, Tamassia, and Goldwasser (2022), I now have a stronger foundation to write efficient, reliable, and maintainable code that follows industry best practices.**

**References**

Goodrich, M., Tamassia, R., & Goldwasser, M. (2022). Data structures and algorithms in Java. Zybooks. ISBN: 9798203227881