4-2 Milestone Three Enhancement Two Algorithms and Data Structure

Christian J. Kesler

Department of Computer Science, Southern New Hampshire University

CS 499: Computer Science Capstone

Professor Joseph M. Conlan

January 30, 2022

Algorithms and Data Structures Artifact

Artifact Description

The artifact I have chosen to showcase my Algorithms and Data Structure capabilities is the Binary Search Tree C++ program. This project was created during CS-260 "Data Structures and Algorithms" over a year ago at SNHU, utilizing a series of C++ methods to generate and manipulate a binary search tree. The nodes within the tree contain an object that holds the data for each entry, along with pointers to the left and right children nodes. This basic structure is utilized after performing brief comparative checks to determine how the nodes ought to be populated. The C++ methods are recursive and efficient, having clear exit cases and minimalistic input required.

Inclusion in Portfolio

This artifact serves to showcase my competency with the C++ programming language, the Microsoft Visual Studio IDE, object oriented programming, data structure and algorithm principles, and observation of programming best practices such as naming conventions and comment etiquette. Considering that C++ is a challenging language with relatively unique syntax and technical complexity used in a variety of mathemtically demanding programs, and that data structures and algorithms are used in staggeringly large number of applications, I would say with confidence that this project is worthy of inclusion in my portfolio.

Enhancement Progress

As with my other artifacts, the first element of this project that I tended to was the comment etiquette. The comments were vague, inconsistent, and sometimes entirely absent throughout this program. I began by adding descriptions for each of the methods, including input clarification and recursive designation. In the main

method, I included comments that describe the nature of the clock computations so that the exact functionality of each case might be easier to discern.

After improving the comments throughout, I turned to expanding or improving on the functionality of the program. I began by including the clock statements across all cases, and improving the readability of said satatements. I then created a destructor for the binary search tree, which is a recursive function that deletes every node in the tree before creating a new empty tree to avoid reading errors. The destructor serves only to pass the root to the method that destroys the entire tree. While I could have simply made a function that destroys the tree from the root and called that, I wanted to leave my methods open ended so that I might use them to delete portions of the tree later on should I need to do so.

Enhancement Reflection

While enhancing this artifact, I realized that the binary search tree is remarkably simple to manipulate and interact with once the recursive functions are handled. The most complex thing that I might try to do besides creating a more dynamic menu interface would be a function that rebalances the tree. This would not be an insurmountable task, but I intend to only focus on something so difficult to develop and test after all my artifacts are in an acceptable state. The other thing that I noticed while enhancing this artifact involved the file that the default data is loaded from. I began to wonder how I might be able to load from a specific file through the menu after program execution, and doubt it would be remarkably complicated. The ability to pass in a file path upon command line execution of the program serves that purpose for now, but I do have ideas regarding future enhancements.