Anaid Rodriguez

CS-499 Computer Science Capstone

Professor Brooke Goggin

Milestone Three: Enhancement Two: Algorithms and Data Structure

04/05/2024

The artifact is a binary search tree (bst) program that reads information from a CSV file and stores that information in ‘bids’ which are there stored in the nodes of the tree. Each bid contains 4 parameters and each node contains 1 bid. The program also includes functions to sort the tree according to InOrder, PostOrder, and PreOrder. This assignment was part of the course CS-300 Data Structures and Algorithms and was originally created and turned in late January of 2023, over a year ago.

I chose this specific artifact for my ePortfolio because I wanted to challenge myself by changing the language of a previous assignment. Additionally, there was some parts of the original program that I wanted to improve because I found them underwhelming and wanted a chance to develop code I can be proud of. This program was ideal because it used sorting algorithms to sort a data structure, it was developed in C++ therefore I had the opportunity to change it to Python, and, lastly, there was a lot of room for improvement that I could work on to fully showcase how much I’ve learned this past year. One of the most important components in the artifact that showcases my skills in algorithms and data structures was writing a program that in best-case scenario would have a time complexity of O(log n) and a time complexity of O(n) in the worst case scenario. The time complexity depends on how balanced the tree is when the program runs (worst case scenario being an unbalanced tree). Being able to develop a program that can run on linear time (or faster) shows my ability to be mindful of speed and execution time. However, it does leave room for improvement as I continue to grow as a computer scientist. Additionally, the efficiency of the program is displayed in the amount of reusable code and the good run time the program has. Finally, the goal of the program is to demonstrate how inserting into a tree and sorting it works. The code accomplishes this and even exceeds the basic needs of the goal by providing 4 different ways to organize the tree as well as executing correctly and quickly. In a nutshell, the program executes correctly, fast, and demonstrates to the user the difference between various sorting algorithms. As for the programming side of it, it is written in such a way that it is fully reusable for any other CSV file that follows the same template as the one in the example.

The artifact was improved in a few areas including making use of the postOrder and preOrder functions, using more in-line comments to explain each function properly, increasing the speed, implementing input validation, increased menu options for the user, and changing the hard coded search and delete functions to user input functions instead. In the original program, the postOrder and preOrder functions were created but were not used anywhere else or in main( ) and the user was not given the option to display the bids in any way other than the InOrder sort. In my enhancement, I not only included the functions in main but I gave the user the option to display the bids in the three different ways while also giving them a fourth option to view the bids in the order they are added to the tree. This allows the user and the programmer to view how the different sorting algorithms work as well as ensuring the programmer that all of the sorting algorithms are working perfectly. Another improvement was the addition of input validation for the main menu. The original artifact does not include validation and made the program crash when anything other than the expected input was typed in. The original artifact also included search and delete functions but these were hard coded so I changed them to allow user input (with input validation) to allow the user to choose what they would like to search for and delete.

The course outcomes I planned on meeting with this enhancement were:

• Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.

• Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.

I met the first goal by developing a program that uses sorting algorithms and coding best practices. The enhanced program is properly commented, with fast and fully efficient algorithms that sort a binary search tree in a few different ways. The second goal was met by developing code that was easy to be read from and worked on by another developer if they were to pick it up. Additionally, the user-end side of the program is engaging and allows the user to interact with it to sort, search, and delete from the bst. One update to my outcome-coverage plan is meeting an additional outcome that I had not planned on meeting originally:

* Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources

While reviewing the artifact originally, I had failed to notice the lack of security it had. While working on this enhancement, though, I found several vulnerabilities, specifically in the lack of user input validation. The practice of proper validation reduces the risks of attacks, such as injections, and is the first line of defense in security.

While I worked on the enhancement, I learned how important proper documentation is because I found some functionality in the artifact that lacked explanation. The main challenge I faced was transferring the code to Python because while I’ve worked with bst’s before, I’ve only ever done so in the C++ language. However, because of this enhancement, I feel so much more confident in my Python skills and my knowledge of the language has increased.