**Enhancement 2: Prompts**

Jesenia Roberts

Southern New Hampshire University

CS - 499: Capstone

Professor Goggin

Jul 28, 2024

**Briefly describe the artifact. What is it? When was it created?**

The chosen artifact was from August of 2023 for the CS-300 for the Data Structures and Algorithms: Analysis and Design course which assisted us in planning, executing, and testing high-quality code. The goal of the project was to read a file that held course IDs, the name of the courses, and their prerequisites, insert the data into a binary search tree, and then have the data be accessible and editable to the user.

**Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?**

This artifact was chosen because the class's purpose was to gain familiarity with data structures, and this project specifically required a deep understanding of navigating data structures to create them. I improved the artifact by adding 3 additional options for the user, 1) to find a successor node when a course is input, 2) to find a predecessor node when a course is input, 3) to have a given range of courses returned when a start course ID and an end course ID are input. I selected this artifact to showcase my ability to evaluate computing solutions when given a scenario, to design and implement these solutions, and to utilize algorithms to write efficient code.

I calculated the time efficiency for each new and existing method and included them in each respective method header summary. Given that many of the functions are recursive it is inherently more difficult for new readers to parse and understand, so for clarity I added additional comments instructing on why each call to recurse was occuring.

I included lines to measure the execution time of various methods in nanoseconds. These were the following measurements:

InOrder (all courses printed): 15442500 ns

InOrderCourses (2 printed): 2931800 ns

InOrderCourses (7 printed): 3801900 ns

Search: 27700 ns

FindSuccessor: 71600 ns

FindPredecessor: 72600 ns

It is worth noting that InOrderCourses is a recursive function with the same iterations as InOrder, but has less execution time because it's not printing all output. However, it goes through the entire tree with the same complexity in both cases. The act of additional prints is likely affecting efficiency.

It is also interesting that FindSucessor and FindPredecessor take longer than Search despite having the same time complexity. It can be surmised that this is because of the extra parameter tracking the successor or predecessor that's being passed in the recursive function.

For the successor and predecessor methods, I could track potential successor and predecessor nodes while traversing the binary search tree for the relevant courseId. This meant that once I encountered it I didn't have to spend additional time looking back through the tree and therefore had an efficiency equivalent to search or O(logn).

**Did you meet the course objectives you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?**

I did meet the course objective I planned with this project which was:

*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.*

I designed and evaluated computing solutions (the provided methods) that solve a given problem (need for additional options to navigate courses) using algorithmic principles and computer science practices (utilizing binary search trees) and standards appropriate to its solution while managing the trade-offs involved in design choices.

In my review, I discussed the possibility of adding more security features but was unable to fit that into the scope. If I had more time I could have removed the recursive element from the project which also would have reduced time complexity.

**Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?**

This artifact was particularly difficult to accomplish. I had to change the initial plan of finding successor and predecessor courses to finding successor and predecessor nodes because I attempted to accomplish the former for a substantial amount of time before realizing that a Binary Search Tree was not the optimal way to retrieve that data. Additionally, for some reason, my IDE (Visual Studio Code) does not work well with C++ and requires some outside tools to make it work, which was also time-consuming. I am certainly more familiar with BSTs now, as well as recognizing where they are not ideal to use.