**Algorithms and Data Structures Enhancement**

Brandon Cook

Southern New Hampshire University

CS 499: Computer Science Capstone

Dr. Pravin Bhandari

September 27, 2025

**Algorithms and Data Structures Enhancement**

This artifact is a program that I developed during my time in CS-300: Data Structures and Algorithms. It consists of a binary search tree (BST) used to sort through a list of courses, creating a catalogue that can be browsed through to learn about a course and its prerequisite courses. The purpose of this project in CS-300 was to display knowledge of data algorithms.

I chose to enhance this project beyond its original capabilities to display the growth in my knowledge of algorithm efficiency and data structures. A standard BST is an effective tool, but it is limited by the structure of the data that is fed into it. BSTs sort items into a tree alphanumerically, so if the list is already alphabetized prior, it can lead to an unbalanced list. The dataset used consists of 50 different courses, meaning that a single search can take as many as 50 comparisons for a single search if the tree was unbalanced. This is not optimal; in larger datasets, optimization is important. The number of node searches in the tree should be limited to the lowest maximum number of searches possible, in order to operate efficiently.

That is why I refactored the BST into an AVL tree, which is a self-balancing BST. Nodes have a balance factor and a height. When added to a tree, the balance factor and height are updated. If the balance factor, which determines if one of the branches in the tree has grown too large, exceeds -1 or 1, the node on the tree is ‘rotated’ to the next available branch, which balances the BST. In this way, a properly implemented AVL tree structure, with a dataset of 50, has a maximum possible number of searches of 7. This result can be visualized by the printed tree (sideways) which displays what the BST looks like after being properly balanced. An optimized tree can search through large datasets quickly, which is important in situations of data analysis or information gathering. For the case here, it could be a useful algorithm to use in a course planner.

While developing this project, I learned the importance of optimization of algorithms as they lead to better search times, especially in larger datasets. Additionally, I learned the value in proper documentation with complex algorithms such as this, as complexity can quickly make the conventions difficult to follow. Finally, I honed my ability to work with recursive methods by building a visualization system that used recursive calls to itself. I find that getting these methods correct to be challenging.

I believe that I have met the course outcomes that I decided on for this enhancement during module one. Those outcomes are listed as follows:

* Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.
* 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.
* Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.

I have built a well-commented algorithm that can be applied to a variety of different scenarios and will allow future developers to easily pick up and continue working on. I have designed a solution using algorithmic principles by improving the efficiency of a BST and making it self-balancing. I have demonstrated an ability to utilize innovative techniques to deliver a valuable solution to a problem.