Brian Voo
Professor Penmatsa
CS 499
23 Nov 2025

4-2 Milestone Three: Enhancement Two: Algorithms and Data Structure

For this milestone, I enhanced my CS-300 Course Planner program to demonstrate deeper algorithmic reasoning and data-structure selection. The original version relied entirely on an unordered_map to store and retrieve course information. While this structure provides efficient average-case lookup time, it did not fully demonstrate my ability to design, evaluate, and compare more complex algorithms and data structures. This made the program a strong candidate for enhancement within the Algorithms and Data Structures category.

To strengthen the artifact, I implemented a full Binary Search Tree (BST) alongside the existing hash table. After loading course data from the CSV file, the program now inserts each course record into both data structures. The BST supports recursive insertion and in-order traversal, allowing the program to output a sorted course list without requiring an additional sorting algorithm. I validated the correctness of the BST by confirming that in-order traversal consistently produced alphabetically sorted output and that search results matched those returned by the hash table.

I restructured the program logic so that searching and display operations leverage the BST while still retaining the hash table for fast direct lookup. This dual-structure approach demonstrates an understanding of algorithm selection and trade-offs, allowing constant-time average lookups with the hash table and ordered traversal with the BST. I tested consistency between the two structures by verifying that course searches returned identical results regardless of which structure was queried.

To ensure proper memory management, I added a cleanup routine to deallocate BST nodes when the program exits. I verified this logic by tracing node deletion paths and confirming that no dangling pointers or orphaned nodes remained. Addressing memory cleanup reinforced the importance of resource management when working with dynamic data structures in C++.

One important trade-off introduced by this enhancement is increased memory usage and synchronization complexity, since course data must be maintained in two structures simultaneously. However, this design choice improves the program's flexibility and better illustrates algorithmic decision-making. Through this enhancement, I gained a stronger understanding of recursion, tree traversal, and the practical implications of managing multiple data structures. Overall, this work demonstrates my ability to analyze existing solutions, apply more advanced algorithms, and evaluate performance and design trade-offs.