CS499: Computer Science Capstone

Milestone Three

Buddy Marcey

buddy.marcey@snhu.edu

Southern New Hampshire University

November 24, 2024

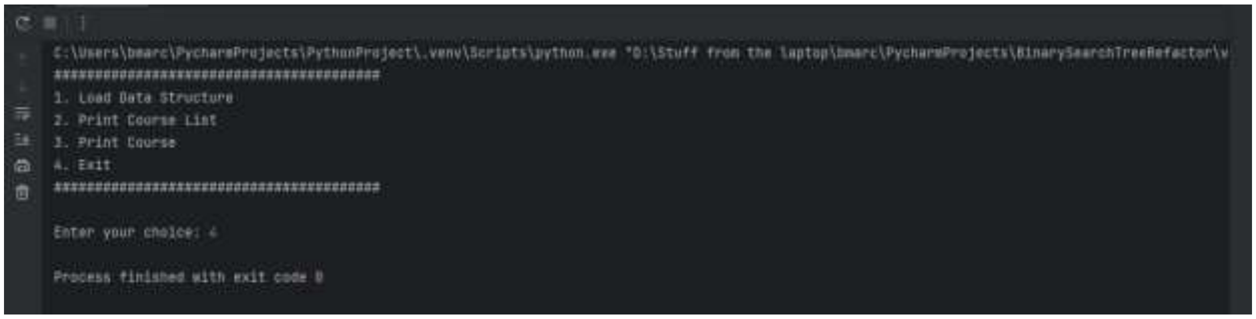
Enhancement Two: Data Structures and Algorithms

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

This artifact was the final project for CS300: Data Structures and Algorithms. It is a CLI tool that imports a file from .csv format and loads the data into a data structure of our choosing; in my case, a binary search tree. I completed this project initially in October 2023.

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 for enhancement because it was the project that I developed with the closest ties to the Data Structures and Algorithms topic. It demonstrates the ability to analyze what needs to be accomplished with the data and build an appropriate structure to realize the end goal. The idea was to build a structure to increase the efficiency of interacting with the data; a binary search tree does search, insert, and delete functions in O(log n) time, which is much more



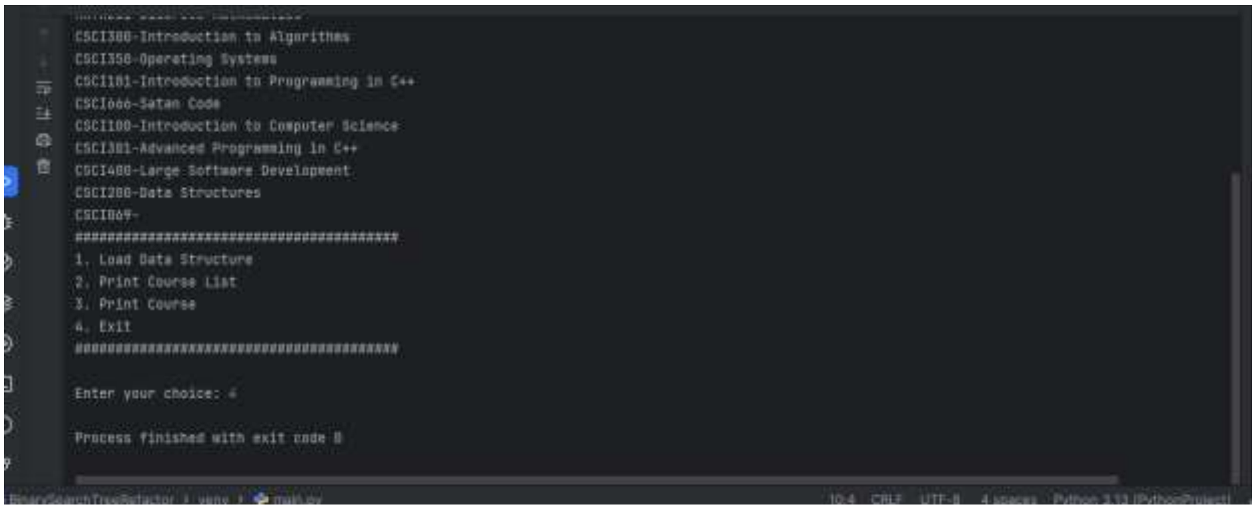
```
C:\Users\bmarc\PycharmProjects\PythonProject\.venv\Scripts\python.exe "D:\Stuff from the laptop\bmarc\PycharmProjects\BinarySearchTreeRefactor\v
##########################################
1. Load Data Structure
2. Print Course List
3. Print Course
4. Exit
##########################################

Enter your choice: 4

Process finished with exit code 0
```

Figure 1: the CLI tool running in Terminal

efficient than other data structures. The CLI tool component gives a more user-friendly way to interact with the data than trying to just read it from a text file manually. The improvement plan involved two components: translating the project into Python to make it more compatible with newer libraries and tools and adapting the binary search tree into a self-balancing Red-Black tree. The former demonstrates the ability to critically analyze and translate code from one language to another, and the latter demonstrates the ability to recognize an inherent software flaw and build an appropriate solution. A generic binary search tree is only a viable data structure if the original data is sufficiently randomized. By changing this to a self-balancing tree, the data can be either sorted or unsorted, as the tree will balance itself at each addition to the tree.



```
CSCI300-Introduction to Algorithms
CSCI350-Operating Systems
CSCI101-Introduction to Programming in C++
CSCI666-Satan Code
CSCI100-Introduction to Computer Science
CSCI301-Advanced Programming in C++
CSCI400-Large Software Development
CSCI200-Data Structures
CSCI069-
########################################
1. Load Data Structure
2. Print Course List
3. Print Course
4. Exit
########################################

Enter your choice: 4

Process finished with exit code 0
```

Figure 2: Another capture of the CLI tool

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

The main proficiency to demonstrate in this module is course outcome 3 (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 (data structures and algorithms)). Developing a system to load data into an appropriate data structure demonstrates this proficiency, and adapting the original binary search tree to a self-balancing tree helps mitigate a design flaw involving datasets that are presorted (a sorted list imported into a standard binary search tree implementation is just one long branch, at which point it could just be a vector or linked list). This enhancement also contributes to the fourth course outcome (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), which was also covered in the previous module. Likewise, the journal submission and this milestone report contribute to the second course outcome (Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts) by demonstrating the ability to present my work.

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?

I learned a lot about how to implement a red-black tree in Python and how logic works to maintain color integrity to keep the tree balanced. During the course, I learned about the benefits of instantiating a Binary Search Tree as a data structure, but my study this week was a good refresher in Big O notation; I barely grasped the concepts from the reading material at the time, but I have a much greater understanding of it now. I had to relearn several Python processes to make the program work. Specifically, in C++ I used two constructor methods, one to initialize an empty object and another to instantiate an object with all parameters included. Python doesn't allow for constructor overloading, so instead I had to implement constructors with default cases. This was not difficult, but it took some time to research how to do it since my Python is a little rusty. My csv parser needs some updates to match the old C++ version of the project; in Python, I implemented the csv functions in the standard library, whereas for the C++ version, I wrote the parser myself. I would like to go through and enhance the parser before submitting it to the portfolio.