Gianmarco Vendramin

Narrative - Enhancement Two: Algorithms and data structures.

This enhancement step continues the work on the original artifact. It has not changed from the previous one because the work will be centered on enhancing this software in various steps, aiming to approach a complete product that can become the starting point for developing a much more complex system.

The selection of this artifact is justified by the fact that it is a good example project that could include all the three enhancements requested. It comes from a course centered around algorithms and data structures, so it is a perfect fit for this milestone enhancement. During the course, multiple search algorithms have been tested with different structures, starting from bubble sort that unfortunately performed poorly with a big O complexity of $O(n^2)$, successively the linear or sequential algorithms have been tested that, however, still performed not exceptionally with $O(n)$. Still, the real final winner was the binary search tree that, at the cost of a bit more complexity and usage of recursive functions, achieved a big O complexity of $O(\log n)$. Preparing and feeding the application with a bigger data set has shown that the binary search tree still performed very well, and no notable degradation in performance has been seen.

The software was improved in multiple parts, and some more menu options have been added to it, like loading the course data from a JSON file, converting a CSV file to JSON format, and emptying the data structure (binary search tree). First of all, an internal function was created to translate the course data coming from the CSV file in JSON format by loading it into a temporary structure or list, creating a JSON schema, and storing the data in the JSON file, adhering to this schema. While loading the data, some code safety functions were implemented to ensure a safe file operation and a sanity check in the loaded data. Moreover, the prerequisites check was replicated to ensure the prerequisite was in the course list.

Another improvement was the addition of a JSON file loading data algorithm that loads the courses directly in the binary search tree without passing through an intermediate structure. It has ensured a more efficient code while keeping the file and coherence checks on the data source. Moreover, to be able to empty the binary search tree without exiting the application, an empty munù function has been added that calls the BinarySearchTree class method "empty." Through this method, the tree's root is set to "None," and the remaining leaves are garbage collected by Python without the need to delete each leaf. All of the above activities are in preparation for the next improvement, where the data will be moved to a database like MongoDB, which will ease the job if we already have the data in JSON format.

With this milestone, I've met the planned outcomes, like the ability to use skills and common tools in computing technologies aligned with industry best practices and keeping a security mindset by ensuring software security in file management and data sanitization. For example, great attention has been put toward checking the data format in the data files by constantly checking the consistency of the data, for example, by checking that each prerequisite is also present as a course inside the courses list. Additionally, some algorithmic principles have been developed to improve the code performance.

While modifying and enhancing the artifact, I've learned how important it is to have an analytic mindset while developing and imagining all the possible things that could go wrong to produce better and safer code. I've also learned that taking the most straightforward path can lead to ineffective and redundant code that degrades its performance in a production environment. One challenge I've faced is how easy it is to create bugs with a dynamic typing environment such as Python coding, where you can easily exchange variable names like "course" with "courses." That led me to spend a lot of time debugging and finding the root cause of the problem.