Andrew Brown

September 29, 2024

Revised: October 15, 2024

CS 499

Algorithms and Data Structure: Milestone Three

The artifact for category two is the course list program from the CS300 class. The program reads a comma delimited file that contains the course number, name and prerequisites. When the user requests the application to display the list of courses, it will sort the list and print out the results.

The reason the program was selected is that it uses a combination of data structures and algorithms to store the course list. The course list has a specific data structure to store the necessary data for each course, such as the course number, course name, and a list of the prerequisites. The course program also uses an algorithm to search for courses and to sort them in alphanumeric order by course number. In the current state, the sorting algorithm uses the built-in sorting that checks over the data as it is sorted, and for the dataset that is commonly used in the application, it will use the quicksort algorithm.

I designed and evaluated 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 by improving the sorting algorithm to use merge sort since the data set for the courses won’t be extremely large. After this change is made, the worst-case time complexity will improve from O(n2) to O(n\*log(n)).

To implement the change in the code, I first needed to figure out how the merge sort algorithm works by looking at the pseudocode and then figuring out how to map it to the vector that was already created. The merge sort algorithm essentially creates temporary vectors and compares a defined key to add the courses to the temporary vector in sorted order. In the pseudocode below, the merge function gets the start, middle, and end indexes of the array and fills the two temporary arrays:A screen shot of a computer program

Description automatically generated

The second important function is the mergeSort function, which recursively calls itself in addition to the merge function to temporarily create and move the values in the main array:

A computer screen with white text

Description automatically generated

The final code that is implemented in the C++ application is very similar to the pseudocode, with additional language-specific code added. In the CourseList class, the original courseSort method was removed and replaced with the two new methods set as private visibility:

A screen shot of a computer code

Description automatically generated

One additional method, called sortCourses, was added, which starts the sorting, and the coursesSorted property was added to keep track of whether the courses were already sorted by the algorithm. From here, the merge function was translated from the pseudocode to C++, and the identifier was also added during the comparison part of the algorithm:A screen shot of a computer program

Description automatically generated

The mergeSort and sortCourses methods were added to handle the recursive calls and start the sorting:

A screen shot of a computer program

Description automatically generated

A black background with blue and yellow text

Description automatically generated

To prevent the vector from being sorted multiple times when the course list is displayed, the application will check the coursesSorted variable and only sort the courses if the operation was not completed. This was implemented in the printCourses method:A screen shot of a computer code

Description automatically generated

The courseSorted variable is set to false whenever a new course is added to the list, such as when the comma-delimited file is loaded.

I Designed, Developed, and Delivered professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts by documenting the changes within the code using inline comments and a header comment along with defining the exact changes in the narrative.

I learned how different sort algorithms can be applied based on the data that needs to be sorted. There are cases where one algorithm is preferred over another based on how the data is structured and the size of the dataset. One challenge that I faced when changing the sort method was setting up the plan of how the pseudocode would be applied in the existing class and how it would successfully sort the vector. The pseudocode assumes that a simple array will be used for sorting, but the CourseList data structure is more complex than the simple array. After figuring out what identifier the algorithm uses to sort, the rest of the code was easy to apply. The other challenge was optimizing the code so as not to pass the array reference as a method parameter since this would be unnecessary with the CourseList class structure. As an industry standard, if a method is part of a class, then it should have visibility to other class members. Once the changes were made, additional testing and debugging were completed to ensure the application worked as expected.