**CS-300 Communication Vector Sorting**

Zachary N. Locke

Southern New Hampshire University

Analysis and Design

Jack Lusby

September 23rd, 2024

My recent project aimed to develop a robust program capable of sorting a dataset of bids extracted from a CSV file, utilizing both selection sort and quicksort algorithms. This initiative reinforced my understanding of sorting mechanisms and underscored their critical role in simplifying data management tasks. By implementing these algorithms, I could transform unstructured data into a well-organized format, facilitating more accessible analysis and decision-making.

I encountered several technical challenges throughout the development, particularly in parsing CSV files. This aspect of the project tested my ability to handle various data types and maintain the integrity of the dataset under processing. To overcome these obstacles, I focused on enhancing my error-handling and debugging skills, ensuring that the program could gracefully manage unexpected inputs or data anomalies. Troubleshooting and refining the error-handling mechanisms significantly improved my proficiency in crafting more resilient software that can reliably perform under diverse conditions.

CS-300 Communication Vector Sorting

Zachary N. Locke

Southern New Hampshire University

Analysis and Design

Jack Lusby

September 23rd, 2024

My recent project is designed to develop a robust application capable of managing and sorting a dataset of computer science courses extracted from a CSV file. This task involves utilizing both selection sort and quicksort algorithms to organize courses alphanumerically by their course codes, which is essential for efficient course management and access.

Technical challenges primarily involved parsing the CSV files to handle various course data formats while maintaining data integrity. Enhancements in error handling and debugging were crucial to manage unexpected data inputs or anomalies effectively. This refinement process was vital in ensuring the application's reliability and performance under varied conditions.

Start

1. Define a structure to hold Course information:

- courseId: string

- courseTitle: string

- coursePrerequisites: list of strings

2. Read CSV file path.

3. Load data from the CSV into a vector of Courses:

- For each row in the CSV file:

- Create a new Course.

- Assign courseId, courseTitle, and coursePrerequisites from the corresponding fields in the row.

- Add the Course to the vector of Courses.

4. Define Sorting Functions:

- Function: SelectionSort(vector of Courses)

- For each position in the vector from the start to the second last:

- Assume the current position has the minimum courseId.

- Compare the courseId at the current position to all courseIds in positions beyond it.

- If a smaller courseId is found, note its position.

- Swap the current position Course with the Course having the minimum courseId found.

- Repeat until all positions except the last have been processed.

- Function: QuickSort(vector of Courses, low index, high index)

- If the segment length (high index - low index + 1) is greater than 1:

- Select a pivot using the median-of-three method based on the 'courseId' attribute.

- Partition the vector into two parts where:

- Courses with courseIds less than the pivot go to the left.

- Courses with courseIds greater than the pivot go to the right.

- Recursively apply QuickSort to the left and right segments.

5. Display Menu in a loop:

- Repeatedly display the following options until the user chooses to exit:

- "1. Load Courses"

- "2. Display All Courses"

- "3. Selection Sort All Courses"

- "4. Quick Sort All Courses"

- "9. Exit"

- Prompt for user input:

- "Enter choice: "

- Read user input and store it in the variable 'choice'.

- Process the choice:

- If choice == 1:

- Load courses from the CSV file.

- Display the count of courses read.

- If choice == 2:

- Display all the loaded courses.

- If choice == 3:

- Invoke the SelectionSort function on the vector of Courses.

- Display the sorted courses.

- If choice == 4:

- Invoke the QuickSort function on the vector of Courses.

- Display the sorted courses.

- If choice == 9:

- "Exit"

- If the input is invalid (not among the listed options):

- Prompt the user with "Invalid choice. Please enter a number between 1 and 9."

End Loop when the user selects "9".

End