CS-300-13175-M01: DSA - Analysis and Design Southern New Hampshire University

Rafael Salazar

February 19, 2025

Final Pseudocode and Runtime Analysis

# Pseudocode

1. File Handling and Data Validation Purpose: Open a file, read its contents, check the data, and handle errors if needed.

Steps:

* Ask the user for the file name (e.g., "courses.txt").
* Try to open the file.
  + If the file cannot be opened, display: "Error: Cannot open file." and stop the program.
* While there is more information in the file:
  + Read a line from the file.
  + Split the line into courseNumber, courseTitle, and prerequisites.
  + If courseNumber or courseTitle is missing, display: "Error: Invalid data format." and stop the program.
* Close the file.
* If any other error happens while reading the file, display: "Error reading file."

1. Creating and Storing Course Objects Purpose: Create course objects to hold course information.

Steps:

* Define a Course object with:
  + courseNumber (text)
  + courseTitle (text)
  + prerequisites (list of text)
* For each valid line of data from the file:
  + Create a Course object.
  + Assign courseNumber, courseTitle, and prerequisites to the Course object.
  + Store the Course object in the data structure (vector, hash table, or binary search tree).

1. Menu Options Purpose: Provide the user with a menu to load data, view the course list, or find course details.

Steps:

* Display the menu options:
  1. Load Course Data
  2. Show All Courses
  3. Show Course Details
  4. Exit
* While the user does not choose Exit:
  1. Ask the user to choose an option.
  2. If the user chooses 1:
     + Load the course data.
  3. If the user chooses 2:
     + Show all courses in alphanumeric order.
  4. If the user chooses 3:
     + Ask the user for a course number.
     + Show the details of that course.
  5. If the user chooses 9:
     + Display: "Goodbye!" and stop the program.
  6. If the user enters anything else:
     + Display: "Invalid choice. Please try again."

1. Show All Courses Purpose: Display a list of all courses sorted by course number.

Steps:

* Sort the courses by courseNumber in ascending order.
* For each course in the sorted list:
  + Display the courseNumber and courseTitle.

1. Show Course Details Purpose: Display detailed information about a specific course.

Steps:

* Ask the user for a course number.
* Search for the course in the data structure.
* If the course is found:
  + Display the courseNumber, courseTitle, and prerequisites.
* If the course is not found:
  + Display: "Course not found."

# Runtime Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | Vector | Hash Table | Binary Search Tree |
| File Read and Object Creation | O(n) | O(n) | O(n) |
| Insert | O(1) (amortized) | O(1) | O(log n) |
| Search | O(n) | O(1) | O(log n) |
| Print Sorted List | O(n log n) | O(n log n) | O(n) |

# Advantages and Disadvantages

## Vector:

* Simple to implement and use
* Good for small datasets
* Inefficient search and insertion

## Hash Table:

* Fast search and insertion (average case)
* Good for fast lookups
* Collisions can degrade performance
* Does not maintain order

## Binary Search Tree:

* Efficient searching and sorting
* Maintains order
* Can become unbalanced (unless self-balancing)
* Insertions/removals can be slower than hash table

# Recommendation

The hash table is recommended due to its fast lookup and insertion times, which align with the advisors' needs for quick course information retrieval. Although it does not maintain order naturally, sorting can be done when printing the course list. This makes the hash table the most efficient choice overall.

Conclusion Each data structure has unique strengths, but the hash table's efficiency in search and insertion operations makes it the optimal choice for the advising program.