Christina Jimenez

Southern New Hampshire University

CS-300

19 June 2024

Project One

**Pseudocode for the Menu**

// Variable to store the user's choice

int choice

// Loop to repeat the menu until the user chooses to exit(case 4)

do

// Display the menu options to the user

print("1. Load data structure")

print("2. Print course list")

print("3. Print course")

print("4. Exit")

// Get the user's choice

choice = input("Enter your choice: ")

// Use a switch statement to perform the appropriate action based on the user's choice

switch (choice)

case 1:

// Load the file data into the data structure

loadDataStructure()

break

case 2:

// Print an alphanumerically ordered list of all the courses in the computer science department

printCourseList()

break

case 3:

// Print the course title and the prerequisites for any individual course

printCourse()

break

case 4:

// Exit the program

break

default:

// Invalid choice

print("Invalid choice. Please try again.")

break

while (choice != 4)

**Pseudocode for printing Computer Science program courses in alphanumeric order**

// Function to sort and print courses in alphanumeric order

Function SortPrintCourses(courses)

// Retrieve all courses from the input data structure

Declare courseList as empty List

// courses havve to be iterable collection of course numbers

For each course in courses

Add course to courseList

End

For

// Sort the list of courses alphanumerically

Sort(courseList)

// Print the sorted list of courses

For each course in courseList

Print(course)

EndFor

EndFunction

**Run Time and Memory**

Vectors Data Structure:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cost Per Line | Times executed | Big O value |
| Opening/ Reading File | 1 | O(n) | O(n) |
| Parsing Line/ Creating Course Object | 1 | O(n) | O(n) |

Hash Table Data Structure :

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cost Per Line | Times executed | Big O value |
| Opening/ Reading File | 1 | O(n) | O(n) |
| Parsing Line/ Creating Course Object | O(1) | O(n) | O(n) |

Tree Data Structure:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cost Per Line | Times executed | Big O value |
| Opening/ Reading File | 1 | O(n) | O(n) |
| Parsing Line/ Creating Course Object | O(log (n)) | O(n) | O(n log (n)) |

Advantages/Disadvanatages of Data Structures:

Vector: Vectors offer simplicity and efficient random access; this makes them straightforward for managing course lists. Vector data structures can dynamically resize, which in turn is ideal for accommodating changing data sizes without manual memory management. However, insertions and deletions can be slow due especially further down the course list due to elements needing to shift. Due to this this data structure can be of high cost.

Hash Table: Hash tables are best at case insertions, deletions, and lookups. This is crucial for dynamic data management, which is ideal for altering and sorting class data. This Hash Table data structure can handle large datasets. However, a con is that hash table do not have inherent order, which re-ordering would be needed to sort through the data. In addition, hash collisions can affect performance.

Tree: Binary Search Trees maintain a sorted order, beneficial for range queries and sorted traversals. The tree data structures are best at searching and retrieving data. Trees can also shrink and grow dynamically which is beneficial for growing data sets. However, some cons of the tree data structure include the following takes large amount of memory to store trees, they are not as flexible as hash tables, and are not good at sorting and grouping data.

Recommendation:

After careful consideration, a hash table is recommended due to its efficient performance for insertions, deletions, and lookups, which are crucial in a dynamic environment like course management, especially when searching for courses and sorting through information such as adding filters for pre-requisites. Although has tables do require additional sorting steps for ordered output, its overall efficiency and flexibility outweigh this drawback, providing optimal performance for the program's requirements compared to other data structures.