Jamie Javis

12/01/2023

CS – 300

Project One

1. Resubmit pseudocode from previous pseudocode assignments and update as necessary.

Vector Sorting

Start

Open the Course Information Document

If file is open

Create an empty set to store course #

Create an empty set to store prerequisite course #

Read each line from the file

While not end of file

Parse the line into courseNumber, courseTitle, and prerequisites

If line is not formatted right

Display error message for file format error

Exit program

If prerequisites not empty

Divide prerequisites into separate course #

For each prerequisite within the divided prerequisites

If prerequisite is not in the set of course numbers then

Display error message that prerequisite does not exist

Exit program

Add courseNumber to set of course #

Add prerequisites to set of prerequisite courses #

Read next line from file

Close the file

Else

Display error message that file could not open

End

Hash Table  
Start

Open the Course Information document

If file is successfully opened

Create empty set to store course #’s

Create empty set to store prerequisite course #’s

Read each line in file

While lines are in file

Remove whitespaces from line

Divide line into multiple parameters

Check if two parameters

Print error message and exit if nonexistent

Pull course information

Pull course #, title, and prerequisites

Check if course # is already in the set of #’s

Print error message and exit if it exists

Add course # to the set course #’s

If prerequisites are present

For each prerequisite

Check if prerequisite is in set course #’s

Print error message and exit if none

Add prerequisite to set of prerequisite course #’s

Create new Course item

Set courseNumber, courseTitle, and prerequisites in Course item

Create Course item to hash table with courseNumber key

Read next line in file

Close the file

Else

Print error message for no file

End

End

Reading File:

Use stream to open file and retrieve information

Open file, if the return value is “-1”, file is not found

Else file is found

While it is not the EOF (End of File)

Read individual lines

IF less than two values in a line, return ERROR

ELSE read parameters

IF third or more parameters exists

IF third or more parameter is in the first parameter elsewhere continue

ELSE return Error

Close file

Create Course Objects Structure:

Initialize Course Structure

Loop file, WHILE not EOF

FOR each line

FOR first and second value

Add course ID and Name

IF a third value exists

Add prereq until newline is found

Create Tree and add Nodes:

Define Binary Tree Class

Create a root to null

Create Insert method

IF Root is Null, current Course is Root

ELSE is course # is less than root, add left

IF left is null, add course #

ELSE

IF course # is less than leaf add left

IF course # is greater than leaf add right

ELSE if the course # is greater than root, add right

IF right is null, add course #

ELSE

IF course # is less than leaf add left

IF course # is greater than leaf add right

Search and Print from Tree:

Receive Input

Construct Print Method

If root not null

Navigate left, output if found

Navigate Right, output if found

2. Create pseudocode for a menu

Print Menu

While running:

User Input

If == 1:

Load Data Structure from File

Display Message

Else If == 2:

Print Alphanumerically Ordered List

Else If == 3:

Input Course ID

Print Course Information

Print Prerequisites if any

Else If == 9:

Print Goodbye Message

Exit Program

Else:

Print Invalid Message

Exit

3. Design pseudocode that will print out the list of the courses in the Computer Science program in alphanumeric order.

Print Alphanumeric Course

Sort Vector Number by alphanumeric order

For each course Vector

Print courseNumber, courseTitle, prerequisites

End

End Program

Print Alphanumeric Course

Create list with Hash Table

Sort Courses by Number by alphanumeric order

For all sorted courses

Print courseNumber, courseTitle, prerequisites

End

End Program

Print Alphanumeric Course

Create list Courses using order from tree

For each sorted course

Print courseNumber, courseTitle, prerequisites

End

End Program

4. Evaluate the run-time and memory of data structures that could be used to address the requirements.

Opening file: Costs 1 since it's opened once regardless of the size.

Checking if file is open: Costs 1 since it's a conditional check (yes or no)

Reading line from file (each line): Costs 1 per line read.

Splitting line into items (each line): Will cost based length of the line.

Creating new Course object (each line): Costs 1 for each Course object made.

Inserting Course object into data structure (per line): Cost varies on the data structure.

* Vectors will cost 1, Hash Table will cost 1 per insertion, Binary Search Tree depends on the size of the input

Closing the file: Costs 1 since it closes once regardless of the size.

5. Considering the strengths and limitations of each data structure, a vector, being one-dimensional, shows beneficial for managing data in this assignment, specifically with the sole focus on courses and file reading. The utilization of a vector also reduces memory usage, although it comes with the disadvantage of being incapable of deleting elements and accommodating multiple data types. In contrast, implementing a hash table in the project offers superior organization and storage capabilities. Accessing information through keys allows for effective retrieval, creation, and deletion of unique elements with organizational features. However, the real disadvantage of hash tables lies in their potential impact on speed due to synchronization processes.

6. Opting for a tree structure improves organization by allowing data to be stored in left or right nodes, facilitating expanded organizing and operations like searches throughout the project. Similar to hash tables, tree structures require longer modification times. In estimating these options, the preference favors using hash tables. Despite the potential trade-off in speed, the organizational benefits are significant. Given the current project focuses on central functionalities like sorting courses, hash tables provide the needed versatility. Moreover, considering potential additions to project functionality in the future, hash tables offer flexibility without complications, making them a suitable choice.

Work Cited

Running Time and Big-O - Learneroo. (n.d.). https://www.learneroo.com/modules/106/nodes/559