Devin Criswell

2/11/2024

Project One

CS 300

Pseudocode

Vector

ENABLE file reading class, fstream

OPEN file

Check file is open

Check file contains data

WHILE file open and contains data

IF two or less values are available

return “error, not found”

ELSE

read the values

IF three or more values are present

IF third value is first

Proceed with query

ELSE

return “error, not found”

Create Course vector for information

Vector<int>courseinfo() created

WHILE the file is open

FOR read file

FOR the values that are first or second

Implement pushback for adding the value to courseinfo

IF more than two values

Implement pushback adding the value to the next line in courseinfo

GET Input from user

WHILE input received

IF input equals a courseNumber

return course info

FOR every prereq listed

return all prereq info

ELSE

Return “error, does not exist”

Hash table

Use fstream for file management

LOAD the file

OPEN the file

If the File is not found

return "error, file not found"

Else the file is found

WHILE reading the file

if the values are less than two

return "error, not found"

else

read the value

if there are three or more values

if the values are first in another location

program continues

else

return "error, not found"

CREATE vector

Course vector nodes

CREATE a hashtable

Class for the hashtable

CREATE method to add items

Implement the method for adding items to the hashtable

WHILE

For

the first value and second value

store the values in a temp node

if there are more than two values

add them to a node

use the created method for adding items on every value

GET an input from user

CREATE

assign a key to the users input

SEARCH

If the key exists

Return information

For every prereq

Return prereq information

Tree

CREATE readCourseFile

Open File

Read file

CREATE validateFileFormat

for

if there less than 2 variables

print("Error: Insufficient parameters on line ")

return false

for

if the prereq is not found as a course

print("Error: Prerequisite not found”)

return false

return true

CREATE CourseObjects

for

course = Course(courseNumber , title)

add course to tree

for

add prereqs to course in the tree

CREATE printCourseInformation

for course in tree

print("Course Number: ", course.courseNumber)

print("Title: ", course.title)

print("Prerequisites: ", course.prerequisites)

Menu

DISPLAY menu

Print “1 Load the file”

Print “2 Course List”

Print “3 Course”

Print “4 Exit”

Print “Make a selection”

INPUT from user

WHILE

Switch

Case 1: load file

Case 2: print course list

Case 3: print course

Case 4: exit program

Alphanumeric Order

CREATE print function for course list

Sort vector

For

Course in the vector

Display the course

CREATE print function for course list

GET values from hashtable

Sort values from hashtable

For

Course from hashtable

Display the course

CREATE print function for course list

Traverse the tree

Sort values from the tree

For

Course from the tree

Display the course

Evaluation

| **Code**  **Vector** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 2 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 2 | n | n |
| **Total Cost** | | | 5n + 1 |
| **Runtime** | | | O(n) |

| **Code**  **Hashtable** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 2 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 1 | 1 | 1 |
| **for each prerequisite of the course** | 2 | n | n |
| **print the prerequisite course information** | 4 | n | n |
| **Total Cost** | | | 9n + 1 |
| **Runtime** | | | O(n) |

| **Code**  **Tree** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 2 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 4 | n | n |
| **Total Cost** | | | 7n + 1 |
| **Runtime** | | | O(n) |

Advantages and Disadvantages

Advantages for Vectors are they are dynamic, they can grow or shrink as needed. Vectors can sort and access data efficiently. They are also fairly easy to learn and are a safe method to use. The disadvantages of Vectors are that they can only store elements of the same data type. They are slower when inserting or deleting elements in the middle of the vector. They also require a small memory overhead due to the ability to resize.

Hash tables are designed to be able to perform fast searches and data retrieval. Hash tables do not require data to be sorted in any order making them an efficient storage type. Since hash tables use key-value pairs it allows them to be dynamic and adaptable for data organization. They can be prone to collisions when two values share the same key, if not corrected it can lead to performance issues. Hash tables require a significant amount of memory since they store the hash table and the data. Since hash tables do not store data in a particular order it can make it difficult to perform certain queries.

Advantages of Trees are that they can be scaled. They can be scaled to store large amounts of data. They use a hierarchical structure to store data. Trees have improved performance because of their ability to search and sort data. Trees use less memory which helps to reduce cost. Trees can have a single point of failure which can make them vulnerable. Trees are not as flexible as other types of data structures. The implementation of a tree can be more difficult than other data structures. Trees can be slower to operate because they have a limited bandwidth.

Recommendation

For this data set I feel that a tree would be the best method for storing the data. The trees’ structure will allow for the data to be searched more easily. The tree will also provide the easiest way to add or delete elements from the structure. Their hierarchical structure works best with a file system or an organizational chart. Their low cost and ability to store large amounts of data is also useful. For these reasons I would choose a tree structure.