Marcus Smith

CS 300

6/10/2023

Project One

**Vector Data Structure Pseudocode**

Fstream is used to open file

Call file to open file, a -1 return value means the file was not found

If file is found

Read file

If course data is less than 2, output error in file

Else read parameter

If course parameter is greater than two, check parameters greater than two, for other location in file

Course Objects:

Course Vector vector<int> courseInfo()

Loop through file

For each line in file, add first two values

Add values to vector

If values are greater than 2, add values until new line

Search Data Structure:

Get course key

Loop through vector

void printCourseInformation(Vector<Course> courses, String courseNumber) {

**for all courses**

**if the course is the same as courseNumber**

**print out the course information**

**for each prerequisite of the course**

**print the prerequisite course information**

**}**

If course key not found in file, print error message

**Hash Table Structure Pseudocode**

Fstream is used to open file

Call file to open, -1 returned means file not found

If file found

Read file

If course data < 2, output error

Else read parameter

If course parameter greater than two, check parameters > 2, for parameters in other locations in file

Course Objects:

Initialize vector nodes

Create insert HashTable function

Loop through file

For values 1 and 2 in file

Place in temp item

If 3rd value exists, add to temp item

Call insert function for all values to insert item into hash table

Search and Print from HashTable:

Take input

Assign key to input

If key is found

Print course information for key

For prerequisites of course

Print prerequisites

**Tree Data Structure Pseudocode**

Fstream used to open file

Call file to open, if -1 returned, file not found

If file found

Read file

If course data < 2

output error

Else

read parameter

If course parameter > 2

Check parameter > 2, for parameters in other locations in file

Course Objects:

Initialize Tree data Structure

Loop file, While file is not empty

For lines in file

For first two values

Add first two values

If line has more than 2 values

Add value till newline found

Search and Print from a Tree Data Structure:

Take input

Assign key to input

Call Search() function

If key not found

Output key was not found

Else

Print keys course information

For prerequisites of course

Print prerequisistes

**Menu Pseudocode**

While(num != 4){

Insert user num

Print( “Menu Options”

“1. Load Data Structure”

“2. Print Course List”

“3. Print Course”

“4. Exit”)

If num is 1:

While not end of file,

read line, add to course temp list

if course parameter > 2, add next parameter till newline

If num is 2:

Create a head node

While(current head != nullptr){

If(current node < node){

swap to end}

if(current node > node){

swap to front}

continue to loop till nodes are place in descending order

}

If num is 3:

Print(“Please enter a course number”)

Enter courseNum

For all courses

If (course == courseNum){

Print course information

For all prerequisites

Print prerequisites

If num is 4:

Print(“Exited the program”)

End program

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

Create a head node

While(current head != nullptr){

If(current node < node){

swap to front}

if(current node > node){

swap to end}

continue to loop till nodes are place in ascending order

}

Print(course list)

**Evaluation**

The programs calls the file using fstream to read the file, and then parses each line and reads the file till it reaches the end. It goes through this to check and see if there are any errors in the file. In this case, for course information, it makes sure there are at least two parameters, and if there is anything less it returns an error in the file.

Binary Tree Chart:

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | Times Executed | Total Cost |
| For all nodes | 1 | n | n |
| If course == courseNum | 1 | n | n |
| Print node information | 1 | 1 | 1 |
| If course has node left | 1 | n/2 | n/2 |
| Print left node, prerequisite course | 1 | n | n |
| If course has node right | 1 | n/2 | 1 |
|  |  | Total Cost | 2(n/2) + 3n + 2 |
|  |  | Runtime | O(n) |

Hash Table Chart:

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | Times Executed | Total Cost |
| For all Courses | 1 | n | n |
| if course == courseNum | 1 | n | n |
| Print course information | 1 | 1 | 1 |
| For course hashtable | 1 | n | n |
| Print prerequisite information | 1 | n | n |
|  |  | Total Cost | 4n + 1 |
|  |  | Runtime | O(n) |

Vector Chart:

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | Times Executed | Total Cost |
| For all courses | 1 | n | n |
| If course == courseNum | 1 | n | n |
| Print course information | 1 | 1 | 1 |
| For prerequisite of course | 1 | n | n |
| Print prerequisite information | 1 | n | n |
|  |  | Total Cost | 4n + 1 |
|  |  | Runtime | O(n) |

When it comes to comes to the three data structures, the believe the best structure is the binary search tree. When comparing their runtime a vector, though it can insert items in fast, when going through the worst run time O(N), it will loop through all the items in the vector. As for the hash table it is possible to insert items in any spot in the table, though even with that, being a better data structure than the vector data structure still does not make it the best out of the three, hash tables can have a hard time having an O(1) complexity. When looking at the binary search tree, it is possible to cut the search in half, even if we go with the worst runtime. Though making a binary tree structure may be the most difficult, going through recursive calls allow for the binary search tree to reduce the time it will take to search a key. With this I would go with a binary search tree.

Resources:

*algorithms - Hash tables versus binary trees*. (n.d.). Computer Science Stack Exchange. <https://cs.stackexchange.com/questions/270/hash-tables-versus-binary-trees>

*zyBooks*. (n.d.). Learn.zybooks.com. Retrieved June 12, 2023, from https://learn.zybooks.com/zybook/CS-300-T5501-OL-TRAD-UG.23EW5/chapter/3/section/15