CS300

Rikki Xaysanasy

Professor Webb

2/25/2024

Project 1 Pseudocode

Open and Read File~

Utilize fstream to open and read file

Initial file open

If file is not successful

Return Error

Else file is successful

While not End of File

Read each line from file

If less than two values in a line

Return Error

Else

Read course number and title

If there is a third or more parameter

For each added parameter

If third of more parameter not found as first parameter in line

Return Error

Close File

Creating Data Structure to Store Course Objects

1. Vector Data Structure~

Validate Course has Prerequisites

If course has Prerequisites

Return course + prerequisites

If no prerequisites

Return course

Create Course Objects

Initialize Course Vector with variables

Read file

While not End of File

Store course object in structure

Searching Vector Data Structure:~

Prompt User for Input

Loop through structure

If user input is the same as course number

Print out course information

If course has prerequisites

Print course information plus prerequisites

Else

Print course information

1. Hash Table Structure:~

For each course in CoursesData

For each prerequisite in course pre requisites

IF prerequisite not in course.

Return Course

Create course Objects

Define course class to store data

Create hastable class

Create method to insert into hashtable

Initialize Loop sequence throughout file

Open file for reading

While not end of file

Read each line

For first and second values

Create temp to hold values

If there is a third value

Add to current

Insert each value

1. Binary Search Tree

Validate prerequisites and add course

For each line in file

For 1st and 2nd value

Add CourseID, CourseName

IF has 3rd value

Add PreREQ

Create course Objects

Create BST class

Create root equal to null

Create insert method

If root is null

Current CourseID is root

Else

If CourseID less than leaf

Add to left sub tree

If coursed greater than leaf

Add to right sub tree

Validate prerequisites and add course

For each line in file

For 1st and 2nd value

Add CourseID, CourseName

IF has 3rd value

Add PreREQ

Creating Menu

Prompt user for input

Print Menu

Output Menu Options “1. Load Course File, 2. Print Course List, 3. Print Individual Course, 4. Exit”

Create switch for choice

Case 1: Load courses

Case 2: Print sorted list

Case 3: Print course Information

Case 4: Break

Printing List In Alphnumerical Order

1. Vector

Create print sorted method alphanumerically

Prompt User for Input for course

While vector is not empty

If input is same as course

Print course

If course has prerequisites

Print course plus prerequisites

1. HashTable

Create method to print course information alphanumerically

Prompt user for Input for course

Assign key to course

If current node equals key

Print course information

If course has prerequisites

Print course + prerequisites

1. Binary Search Tree

Create method to print course information alphanumerically

Prompt user for input for course

Current node equal to root

While current root is not null

If course equals current

Print course information

If course has prerequisites

Print course + prerequisites

Evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| Vector Code | Line Cost | Executions | Total Cost |
| Create Vector Method | 1 | 1 | 1 |
| Each Line in File | 1 | n | n |
| Create course item | 1 | n | n |
| Check for PreReq | 1 | n | n |
| Append for PreReq | 1 | n | n |

Total Cost = 4n+1

Runtime = O(n)

|  |  |  |  |
| --- | --- | --- | --- |
| HashTable Code | Line Cost | Executions | Total Cost |
| Create Hash Table | 1 | 1 | 1 |
| Each Line in File | 1 | n | n |
| Create temp to hold values | 1 | n | n |
| Check for PreReq | 1 | n | n |
| Append for PreReq | 1 | n | n |
| Insert value | 1 | n | n |
|  |  |  |  |

Total Cost = 5n+1

Runtime = O(n)

|  |  |  |  |
| --- | --- | --- | --- |
| BinaryTree Code | Line Cost | Executions | Total Cost |
| Create BST Class | 1 | 1 | 1 |
| Create root equal Null | 1 | 1 | 1 |
| Create Insert Method | 1 | 1 | 1 |
| For Each Line in File | 1 | n | n |
| If node is less then root append to left | 1 | n | n |
| If node is greater than root append right | 1 | n | n |
| Check for PreReq | 1 | n | n |
| Append for PreReq | 1 | n | n |

Total Cost = 5n+3

RunTime = O(n)

The data structures that were implemented have their pros and cons with which structure would be best to utilize. A vector is simple and efficient being the fastest method when reading a file and adding course objects, effectively making this method the most straight forward.

A hash table has an average runtime that closely resembles the binary search tree method, but a has table efficient for searching for courses by their coursed numbers.

A binary search tree maintains all its elements in a more sorted order which is generally more organized. Even though this method could be a little bit complex to implement between the other two. This method is average along with the hash table method when it comes to delivering the required information in a timely manner.

Overall, I would like to implement a vector structure for this project. A vector can quickly sort and read a file and print out the required information quickly. Even though a vector loses time during the search, I still believe it’s a good viable option compared to the other two methods.