Hunter Miller

Project One

8/16/2025

Psuedocode:

**Vector Sorting:**

Opens file, reads data from file, parses line, checks for file format errors:

Use fstream to open file information.

Read each line of the file.

Check each line for parameters.

**If** the line has two or more parameters

**Parse** parameters and store information

Check if prerequisite exist as course in file

**If** a prerequisite does not exist within the file, proceed to the next line.

Create new course using parsed information

Store object in vector

Loop until the file has been completed.

Create Course Objects:

Initialize course vector vector<int>courseInfo()

Open file

**While** not EOF

For line in file

For values one and two

Use pushback for vector value

**If** more values exist

Use pushback until new line

Search for Data Structure:

Ask for user input (course number)

Search vector for match

**If** match is found

**Print** course information and prerequisites

**Else** return error

**Hash Table:**

CREATE Course Structure

CREATE Hash Table for Course Objects

LOAD File

OPEN File

IF Return –1

PRINT Error

ELSE File Opened Successfully

WHILE Not at end of file

READ Line from file

SPLIT line into tokens

IF less than 2 values

RETURN Error

ELSE Continue

CREATE Course Object

SET courseNumber and title from tokens

IF number of tokens is greater than 2

FOR each token after the first 2:

IF the token is not found as course in file:

PRINT Error

ELSE

ADD token to prerequisites

ADD course to HashTable

CLOSE File

PRINT HashTable

FOR each Course in HashTable

PRINT courseNumber, title, prerequisite

END

**Binary Search Tree:**

Use fstream to open file

Call to open file

IF return –1, output “Error”

ELSE, file opened successfully

WHILE not end of file

Read each line of file

IF there are less than 2 values, return “Error”

ELSE continue to read file

IF there is a 3 or more parameters

IF third or more parameter is in first parameter somewhere else

Return “Error”

Close file

Initialize Course structure

LOOP through file

WHILE not end of file

For each line in file

For first and second values

ADD courseID, courseName

IF a third value exists

ADD prerequisite until newline found

DEFINE Binary Tree Class

Create root pointing to null

Create insert method

IF root is null, current course is root

ELIF course number is less than root, add left

IF left is null, add course number

ELSE

IF course number is less than leaf, add left

IF course number is greater than leaf, add right

ELSE if course number is greater than root, then add right

IF right is null, add course number

ELSE

IF course number is less than leaf, add left

IF course number is greater than leaf, add right

Ask for user input

Create print method

IF root is not null

Traverse left, output if found

Traverse right, output if found

ELSE output “Error, nothing found”

**Menu:**

Set choice to 0;

Create while loop for menu

WHILE choice does not equal 4

Output Menu choices;

Create switch

Case 1: loadCourses

Case 2: printSorted

Case 3: printCourseInfo

Case 4: End program

**Print Sorted List:**

Create method to print sorted list

Create partition method

Set lowIndex to first element

Set highIndex to last element

Set pivot to midpoint

Decrement highIndex while pivot < highIndex

Swap lower values to left of pivot, higher to right

Set temp value to lowIndex

Set lowIndex to highIndex

Set highIndex to temp value

Create sorting method

Set mid to 0, lowIndex to first, highIndex to last

If first value >= last value, return

Set lowEndIndex to partition

Make recursive call to sorting

Create display course method

Loop through vector

Create inOrder method

|  |  |  |  |
| --- | --- | --- | --- |
| Vector | Line Cost | Times Executed | Total Cost |
| Create Vector | 1 | 1 | 1 |
| Open and read file loop | N | 1 | N |
| Create course item | 1 | N | N |
| While prerequisites exist | 1 | N | N |
| Append prerequisites | 1 | N | N |
| Pushback course item | 1 | N | N |
| Total Cost | | | 5N+1 |
| Runtime | | | O(n) |

|  |  |  |  |
| --- | --- | --- | --- |
| Hash Table | Line Cost | Times Executed | Total Cost |
| Create hash table | 1 | 1 | 1 |
| Insert method | 0 | 0 | 0 |
| Create key for course | 1 | n | n |
| If not entry found for key | 1 | n | n |
| Assign node to key | 1 | n | n |
| Else | 1 | n | n |
| Assign old node to UINT\_MAX, set to key, set old node to course, and old node next to nullptr | 4 | n | 4n |
| Else | 1 | n | n |
| Find next node | 1 | n | n |
| Add new newNode to end | 1 | n | n |
| For each line in file | 1 | n | n |
| Create vector course item | 1 | n | n |
| While prerequisite exists | 1 | n | n |
| Append prerequisite | 1 | n | n |
| Insert course item | 1 | n | n |
| Total Cost | | | 16n+1 |
| Runtime | | | O(n) |

|  |  |  |  |
| --- | --- | --- | --- |
| Tree | Line Cost | Time Executed | Total Cost |
| Create Tree | 1 | 1 | 1 |
| Add node method | 0 | 0 | 0 |
| If root is null, add root | 1 | 1 | 1 |
| If node is less than root, add left | 1 | n | n |
| If no left node exists | 1 | n | n |
| Node becomes left node | 1 | n | n |
| If node is greater than root, add right | 1 | n | n |
| If no right node | 1 | n | n |
| Node becomes right node | 1 | n | n |
| For each line in file | 1 | n | n |
| Create vector course item | 1 | n | n |
| While prerequisite exists | 1 | n | n |
| Append | 1 | n | n |
| Insert Course item | 1 | n | n |
| Total Cost | | | 11n+2 |
| Runtime | | | O(n) |

Advantages and disadvantages of Vectors:

**Advantages:**

* **Fast method for reading the file**
* **Fast method for adding course objects**

**Disadvantages:**

* **Not great for searching for a specific course**

Advantages and disadvantages of Hash Tables:

**Advantages:**

* **Can search through a list quickly**

**Disadvantages:**

* **A hash table cannot be sorted; the data must first be extracted**
* **Slower**

Advantages and disadvantages of Binary Search Trees:

**Advantages:**

* **Can quickly sort data**
* **Quicker than a vector**

**Disadvantages:**

* **Slower**

Creating a vector for the project would be the quickest way to sort and print the data. A Hash table would not be best because of the information needing to be extracted and would take longer.