Sarah Giannini

CS 300

2/16/25

Project 1

**1. Design pseudocode to define how the program opens the file, reads the data from the file, parses each line, and checks for file format errors.**

Open file

Initialize vector<Course> courses.

for each line in the file:

split the line by commas to get course number, title, and prerequisites.

if the line has fewer than two parameters:

print error

continue

courseNumber = first value

courseTitle = second value

Create a new course object with courseNumber and courseTitle.

if there are prerequisites:

store prerequisites in the course object.

Add the course object to the courses vector.

Close file

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

for all courses

if the course number matches courseNumber

-print the course title

for each prerequisite of the course

-print the prerequisite course information

}

**Runtime analysis:**

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Execute | Total Cost |
| Open File | 1 | 1 | 1 |
| Initialize vector<Course> courses | 1 | 1 | 1 |
| For each line in the file: | 1 | N | N |
| Split the line by commas to get course number,title,and prerequisites | 1 | N | N |
| if the line has fewer than two parameters: | 1 | 1 | 1 |
| Print error | 1 | 1 | 1 |
| Continue | 1 | 1 | 1 |
| Course Number = first value | 1 | 1 | 1 |
| Course title = second value | 1 | 1 | 1 |
| Create a new course object with course number and course title | 1 | 1 | 1 |
| If there are prerequisites | 1 | N | N |
| store prerequisites in the course object. | 1 | N | N |
| Add the course object to the courses vector. | 1 | N | N |
| Close file | 1 | 1 | 1 |
|  | Total Cost |  | 5n+9 |
|  | Run time |  | O(n) |

2. **Design pseudocode to show how to create course objects and store them in the appropriate data structure.**

Open file

Initialize HashTable<Course> courses

for each line in the file:

split the line by commas to get course number, title, and prerequisites.

if the line is invalid:

print error

continue

courseNumber = first value

courseTitle = second value

Create a new course object with courseNumber and courseTitle.

if there are prerequisites:

store prerequisites in the course object.

Add the course object to the hashtable with courseNumber

Close file

void searchCourse(HashTable<Course> courses, String courseNumber) {

if the course exists in hash table using courseNumber

-print course title

for each prerequisite of the course

-print the prerequisite course information

else

- print not found.

}

**Runtime analysis:**

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Execute | Total Cost |
| Open File | 1 | 1 | 1 |
| Initialize HashTable<Course> courses | 1 | 1 | 1 |
| For each line in the file: | 1 | N | N |
| Split the line by commas to get course number,title,and prerequisites | 1 | N | N |
| if the line is invalid: | 1 | 1 | 1 |
| Print error | 1 | 1 | 1 |
| Continue | 1 | 1 | 1 |
| Course Number = first value | 1 | 1 | 1 |
| Course title = second value | 1 | 1 | 1 |
| Create a new course object with course number and course title | 1 | 1 | 1 |
| If there are prerequisites | 1 | N | N |
| store prerequisites in the course object. | 1 | N | N |
| Add the course object to the courses hashtable. | 1 | N | N |
| Close file | 1 | 1 | 1 |
|  | Total Cost |  | 5n+9 |
|  | Run time |  | O(n) |

**3. Design pseudocode that will search the data structure for a specific course and print out course information and prerequisites**

Open file

Initialize Tree<Course> courses

for each line in the file:

split the line by commas into course number, title, and prerequisites.

if the line is invalid:

print error

continue

courseNumber = first value

courseTitle = second value

Create a course object with courseNumber and courseTitle.

if there are prerequisites:

store them in the course object.

Insert the course object into the tree using courseNumber.

Close file

void searchCourse(Tree<Course> courses, String courseNumber) {

start at root of the tree

 if the course number matches courseNumber

-print course title

for each prerequisite of the course

-print the prerequisite course information

else if courseNumber is less than the current node's course number

- move to the left child

else

-move to the right child

-print not found.

}

**Runtime analysis:**

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Execute | Total Cost |
| Open File | 1 | 1 | 1 |
| Initialize Tree<Course> courses | 1 | 1 | 1 |
| For each line in the file: | 1 | N | N |
| Split the line by commas to get course number,title,and prerequisites | 1 | N | N |
| if the line is invalid: | 1 | 1 | 1 |
| Print error | 1 | 1 | 1 |
| Continue | 1 | 1 | 1 |
| Course Number = first value | 1 | 1 | 1 |
| Course title = second value | 1 | 1 | 1 |
| Create a new course object with course number and course title | 1 | 1 | 1 |
| If there are prerequisites | 1 | N | N |
| store prerequisites in the course object. | 1 | N | N |
| Insert the course object into the tree using courseNumber. | 1 | N | N |
| Close file | 1 | 1 | 1 |
|  | Total Cost |  | 5n+9 |
|  | Run time |  | O(n) |

4. **Design pseudocode for the menu functionality to load file data and display the courses.**

Display Menu

Option 1

open file Initialize selected data structure

for each line in the file

split the line into course number, title, and prerequisites

create course object and add it to the data structure

Close file

Option 2

sort the courses in the data structure by course number.

print the sorted list to the display.

Option 3:

ask user to enter course number

search the data structure for the course and print the title and prerequisites.

Option 9:

exit the program

**5. Design pseudocode to print out the list of the courses in alphanumeric order (sorted by course number).**

Sort the courses in the data structure by course number in alphanumeric order

For vector

use sorting algorithm

make sure sorting is done based on the course number in alphanumeric order

For hash table

convert the hash table keys (course numbers) into a list

sort the list of course numbers in alphanumeric order

iterate through the sorted list and use the course number to print the corresponding course

For tree

perform an in-order traversal of the tree, which visits nodes in alphanumeric order

print each course while traversing the tree.

Print the sorted list of courses to the display

display the course number and title for each course in the sorted order.

**Analysis:**

In this project, I learned that each data structure has its strengths and weaknesses. Vectors are simple and easy to use, making them great for small datasets, but searching through them can be slow because it requires looking at each course one by one. Hash tables are fast for finding specific courses, but they don’t keep the data in a particular order, so you would need to sort the courses after loading them. Binary search trees keep the data sorted, which makes it easier to print the courses in order, but adding courses to a tree takes more time compared to vectors or hash tables. This would mean that if sorting is most important, binary search trees are the better choice, but hash tables are better for quickly finding a course. I think that if you need both fast searching and sorted data, a binary search tree is likely the best option.