Sean Glass

CS-300: DSA: Analysis and Design

Module Five Milestone

August 6, 2023

**Pseudocode from previous assignments:**

// Pseudocode vector data structure

function loadCourseData(filename):

courses = empty Vector<Course>

open file with filename for reading

if file cannot be opened

print error message and return empty courses Vector

for each line in the file

tokens = split line by whitespace

if tokens.length < 2

print error message for file format error

continue to next line

courseNumber = tokens[0]

courseTitle = tokens[1]

prerequisites = empty Vector<String>

for i = 2 to tokens.length - 1

prerequisite = tokens[i]

prerequisites.add(prerequisite)

course = create Course object with courseNumber, courseTitle, and prerequisites

courses.add(course)

close the file

return courses

// Pseudocode for printing course information using a vector data structure

function printCourseInformation(courses, courseNumber):

found = false

for each course in courses

if course.courseNumber is equal to courseNumber

found = true

print course.courseNumber

print course.courseTitle

if course.prerequisites is not empty

print "Prerequisites:"

for each prerequisite in course.prerequisites

print prerequisite

break

if not found

print "Course not found."

// Pseudocode for hash table data structure

// Open file, read the data, and check for file format errors

Function readValidateFile(filePath):

Declare an empty hash table

// Try / catch statement to catch any errors

Try:

Open the file in read mode

For each line in the file:

Parse the line to extract course number, title, and prerequisites

If course number, title, and at least one prerequisite exist:

For each prerequisite:

If prerequisite is not present in hash table:

Print an error message indicating that the prerequisite course is not found in the file

Close the file

Return

Create a new course object with course number, course title, and prerequisites

Insert the course object into hash table using course number as the key

Else:

Print an error message indicating that the line is not formatted correctly

Close the file

Return

Close the file

Return the hash table

Catch any file-related errors:

Print an error message indicating that there was an issue with the file

Return

// create a course object and store it in the hash table

Function createCourseObject(courseNumber, courseTitle, prerequisites):

Declare a new course object

Set the course number, course title, and prerequisites instance variables of the course object

Return the course object

// Print course information and prerequisites using the hash table

Function printCourseInfo (hashTable):

For each key-value pair (courseNumber, courseObject) in hash table:

Print "Course Number: ", courseObject.courseNumber

Print "Course Title: ", courseObject.courseTitle

If courseObject.prerequisites is not empty:

Print "Prerequisites: ", courseObject.prerequisites

Print a newline

// Pseudocode for search tree data structure

Declare coursesVector as vector of Course objects

Try:

Open file for reading

Read each line from the file

While line is not empty:

Parse the line to extract course number, title, and prerequisites

If (course number is empty or title is empty or (prerequisites is not empty and there are less than two parameters on the line)):

Print error message

Else:

For each prerequisite:

If prerequisite is not found in any course number:

Print error message

Exit the loop

Create a new Course object with course number, title, and prerequisites

Add the Course object to courses vector

Read the next line from the file

Close the file

Catch FileNotFound or IOException exception:

Print error message

// Print course information and prerequisites

For each course in courses vector:

Print "Course Number: " + course number

Print "Title: " + course title

If course prerequisite exists:

Print "Prerequisites:"

For each prerequisite:

Print "- " + prerequisite

Print newline

End function

**Menu Pseudocode:**

Load file data:

Try:

Open file for reading

Read each line from the file

While line is not empty:

Parse the line to extract course number, title, and prerequisites

If course number is not empty and title is not empty:

Create a new course object with course number, title, and prerequisites

Add the Course object to course vector

Read the next line from the file

Close the file

Catch errors:

Print error message

End function

Print alphanumeric course list:

Sort course vector by course number

For each course in vector:

Print course number + ": " + course title

Print newline

End function

Print course information:

Create Found variable and set it to False

For each course in vector:

If course number is equal to course number:

Print "Course Title: " + title

If course prerequisite(s) exists:

Print "Prerequisites: "

For each prerequisite:

Print "- " + prerequisite

Found = True

Exit the loop

If Found is False:

Print "Course not found."

End function

Main menu function:

Declare a vector of Course objects

Repeat:

Print "Main Menu:"

Print "1. Load Data Structure"

Print "2. Print Course List"

Print "3. Print Course"

Print "4. Exit"

Input choice

If choice is 1:

Call load file data function

Else if choice is 2:

Call print alphanumeric course list function

Else if choice is 3:

Print "Enter Course Number: "

Input course number

Call print course information function

Else if choice is 4:

Exit the program

Else:

Print "Invalid choice."

Until choice is 4

End function

Call main menu function

**Runtime analysis chart**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Vector | Hash Table | Binary Search Tree |
| Reading the file | O(n) | O(n) | O(n) |
| Creating course objects | O(n) | O(n) | O(n log n) |

**Advantages and disadvantages**

Each data structure has its own advantages and disadvantages when it comes to runtime analysis. Some of these advantages and disadvantages of each data structure are as follows:

* Vector
  + Advantages
    - Simple and efficient for linear data storage
    - Quick access by index
  + Disadvantages
    - Insertions and deletions can be slow
      * Dealing with elements in the middle of the vector require multiple shifts
* Hash Table
  + Advantages
    - Fast access and insertion
    - Great for large amounts of data
  + Disadvantages
    - Could create hashing collisions
    - Not ordered
* Binary Search Tree
  + Advantages
    - Ordered
    - Great when you need sorted data
    - Balanced trees are more efficient
  + Disadvantages
    - Slower insertions compared to hash tables
    - If tree becomes unbalanced, performance is affected

**Recommendation:**

Based on the analysis of each of these data structures, I would recommend using a hash table to implement the code. A binary search tree could also get the job done, but the hash table is more consistent in its runtime. Although, if keeping an ordered list of courses is required, the I would actually recommend a binary search tree, despite the slightly slower insertions.