Project One

Justin Murray

Vector Pseudocode

open file

for all lines in the file:

if line does not contain at least two parameters:

print error

return

else if line contains more than two parameters:

print error

Return

else:

course number, course title, prerequisites = parse line(line)

if prerequisites are not empty:

for each prerequisite in prerequisites:

if prerequisite does not exist in the file:

print error

Return

close the file

catch (FileOpenException e):

print error

function parse line

split line into components using comma as delimiter

return course number, course title, prerequisites if any, otherwise return empty list

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

Define data structure for a course with course number, course title, and prerequisites list

Define an empty list called courses list

Open file

For each line in the file:

Extract the course number, course title, and prerequisites from the line

Create a new course object

Set the course number and course title of the course object

Add each prerequisite to the prerequisites list of the course object

Add the course object to the courses list

Close the file

Return the courses list

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

function search and print course (courses list, target course number)

found = false

for each course in courses List:

if course number is equal to target Course Number:

print "Course Number: " + course number

print "Course Title: " + course titl

print "Prerequisites:"

for each prerequisite in prerequisites list:

print "- " + prerequisite

else:

print no prerequisites for this course.

found = true

Break

if not found

print course not found in the list

Hash Table Pseudocode:

Start Program

Open and read file

Loop each line.

Parse each line into course name, course number, and course prerequisites

Look up file format errors

If found

Error message Display: “File not formatted properly.”

End program

Look up the course title

IF totalPrerequisites is found, Look up for course number

add to array

IF course type parameters are less than two

Skip course type

ELSE

Add course name, course number, & course prerequisites to the hash table

IF course prerequisites exist

check to see if course prerequisites are before the course add to the hash table

IF course prerequisites are not found

skip course

display error msg: “Course not found.”

Create a function to Print Course information

If any Print the course name. Course number and course prerequisites

Print Schedule sample

Loop through each course in the vector

Print course info (course name, Course number, and course prerequisites)

Add total count for course prerequisites

Return Total Prerequisites

Get all course number list

Sort the list

Print the sorted list of course number

Create a function to print out the info on a course and its Prerequisites

If course null

Print Course Info

Print All

Binary Table Psuedocode

. Create a List for: courseNumber, courseTitle, prerequisites.

2. Create a BinarySearchTree class with methods to insert and search nodes.

3. Initialize an empty BinarySearchTree named `courseTree`.

4. Try opening file:

try

Open file "courses.txt" for reading

catch any IOErrors:

Print "Error: Unable to open file."

Exit program

5. Read the file:

for each line in file:

Strip any leading/trailing whitespace from the line

If the line is empty or starts with a comment character (# or //):

Continue to the next line

Split the line into components based on a delimiter (e.g., comma, tab):

course\_data = split(line, delimiter)

If the number of components is less than 2:

Print "Error: Insufficient data in line [line number]."

Continue to the next line

Extract courseNumber, courseTitle from courseData

Create an empty list for prerequisites

If there are more components (indicating prerequisites are provided):

Extract and split prerequisites string, add them to prerequisites list

For each prerequisite in prerequisites list:

If prerequisite not found in courseTree:

Print "Error: Prerequisite [prerequisite] not found in file."

Continue to the next line

Create a Course object with courseNumber, courseTitle, and prerequisites list.

Insert the Course object into courseTree using courseNumber as the key.

6. Close the file

7. End

1. Define a function `printCourse(course)`:

- Print course.courseNumber, course.courseTitle

- If course.prerequisites is not empty:

Print "Prerequisites: "

for each prerequisite in course.prerequisites:

Print prerequisite

- Print a newline for separation

2. Define a function `printCourseTree(courseTree)`:

- Perform an in-order traversal of `courseTree`:

If current node is not null:

Recursively print left subtree

Print current node (call `printCourse(currentNode.data)`)

Recursively print right subtree

3. Call `printCourseTree(courseTree)` to print all course information stored in the binary search tree.

# Run Time Analysis

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Vector** | **Hash Table** | **Binary Tree** |
| **Loading Data** | O(1) | O(1) – O(N)  *\*depends on if there are collisions* | O(log N) |
| **Search** | O(n) | O(1) – O(N)  *\*depends on if there are collisions* | O(log N) – O(N)  *\*depends on balance of the tree* |
| **Sort/Print** | O(N log N) *\*using quick sort* | O(N)  *\*assumes the table is created in order* | O(N)  *\*in order traversal* |

Deciding which data structure to choose depends on how the data will be accessed and how frequently. If the data only needs to be loaded infrequently there are no advantages after the initial load. If the data needs to be searched often the hash table could be better than the binary tree assuming an efficient and well-designed hash function or a very unbalanced tree.