Reginald Cooper CS300

Instructor Level 6/15/2024

Vector Data Structure

1. Opening the File, Reading the Data, Parsing Each Line, and Checking for Formatting Errors

function openFile(fileName):

file = open fileName

while file has next line:

line = read next line from file

courseData = split line by comma

if courseData length < 2:

print "Formatting error in line: " + line

else:

courseNumber = courseData[0]

courseName = courseData[1]

prerequisites = courseData[2:]

createCourse(courseNumber, courseName, prerequisites)

close file

2. Creating Course Objects

function createCourse(courseNumber, courseName, prerequisites):

course = new Course

course.number = courseNumber

course.name = courseName

course.prerequisites = prerequisites

add course to vector courses

3. Printing Course Information and Prerequisites

function printCourse(courseNumber):

for course in courses:

if course.number == courseNumber:

print "Course: " + course.number + " - " + course.name

for prereq in course.prerequisites:

print "Prerequisite: " + prereq

4. Menu Options

function displayMenu():

print "1. Load Data"

print "2. Print All Courses"

print "3. Print Course Information"

print "9. Exit"

function handleMenuSelection(selection):

if selection == 1:

openFile("courses.txt")

elif selection == 2:

printAllCourses()

elif selection == 3:

courseNumber = input "Enter course number: "

printCourse(courseNumber)

elif selection == 9:

exit program

Printing All Courses in Alphanumeric Order

function printAllCourses():

sortedCourses = sort courses by course.number

for course in sortedCourses:

print "Course: " + course.number + " - " + course.name

Runtime Analysis for Vector Data Structure

| Code | Line Cost | # Times Executes | Total Cost |

|------------------------------------|-----------|------------------|------------|

| for all courses | 1 | n | n |

| if the course is the same as courseNumber | 1 | n | n |

| for each prerequisite of the course | 1 | 1 | 1 |

| print the prerequisite course information | 1 | n | n |

| Total Cost | | | 4n + 1 |

| Runtime | | | O(n) |

Hash Table Data Structure

1. Opening the File, Reading the Data, Parsing Each Line, and Checking for Formatting Errors

function openFile(fileName):

file = open fileName

while file has next line:

line = read next line from file

courseData = split line by comma

if courseData length < 2:

print "Formatting error in line: " + line

else:

courseNumber = courseData[0]

courseName = courseData[1]

prerequisites = courseData[2:]

createCourse(courseNumber, courseName, prerequisites)

close file

2. Creating Course Objects

function createCourse(courseNumber, courseName, prerequisites):

course = new Course

course.number = courseNumber

course.name = courseName

course.prerequisites = prerequisites

add course to hashTable with key course.number

3. Printing Course Information and Prerequisites

function printCourse(courseNumber):

course = hashTable get courseNumber

if course is not null:

print "Course: " + course.number + " - " + course.name

for prereq in course.prerequisites:

print "Prerequisite: " + prereq

4. Menu Options

function displayMenu():

print "1. Load Data"

print "2. Print All Courses"

print "3. Print Course Information"

print "9. Exit"

function handleMenuSelection(selection):

if selection == 1:

openFile("courses.txt")

elif selection == 2:

printAllCourses()

elif selection == 3:

courseNumber = input "Enter course number: "

printCourse(courseNumber)

elif selection == 9:

exit program

5. Printing All Courses in Alphanumeric Order

function printAllCourses():

sortedCourses = sort hashTable values by course.number

for course in sortedCourses:

print "Course: " + course.number + " - " + course.name

Runtime Analysis for Hash Table Data Structure

| Code | Line Cost | # Times Executes | Total Cost |

|------------------------------------|-----------|------------------|------------|

| for all courses | 1 | n | n |

| if the course is the same as courseNumber | 1 | 1 | 1 |

| for each prerequisite of the course | 1 | 1 | 1 |

| print the prerequisite course information | 1 | n | n |

| Total Cost | | | 2n + 2 |

| Runtime | | | O(n) |

Binary Search Tree Data Structure

1. Opening the File, Reading the Data, Parsing Each Line, and Checking for Formatting Errors

function openFile(fileName):

file = open fileName

while file has next line:

line = read next line from file

courseData = split line by comma

if courseData length < 2:

print "Formatting error in line: " + line

else:

courseNumber = courseData[0]

courseName = courseData[1]

prerequisites = courseData[2:]

createCourse(courseNumber, courseName, prerequisites)

close file

2. Creating Course Objects

function createCourse(courseNumber, courseName, prerequisites):

course = new Course

course.number = courseNumber

course.name = courseName

course.prerequisites = prerequisites

add course to binary search tree

3. Printing Course Information and Prerequisites

function printCourse(courseNumber):

course = bst get courseNumber

if course is not null:

print "Course: " + course.number + " - " + course.name

for prereq in course.prerequisites:

print "Prerequisite: " + prereq

4. Menu Options

function displayMenu():

print "1. Load Data"

print "2. Print All Courses"

print "3. Print Course Information"

print "9. Exit"

function handleMenuSelection(selection):

if selection == 1:

openFile("courses.txt")

elif selection == 2:

printAllCourses()

elif selection == 3:

courseNumber = input "Enter course number: "

printCourse(courseNumber)

elif selection == 9:

exit program

5. Printing All Courses in Alphanumeric Order

function printAllCourses():

sortedCourses = bst in-order traversal

for course in sortedCourses:

print "Course: " + course.number + " - " + course.name

Runtime Analysis for Binary Search Tree Data Structure

| Code | Line Cost | # Times Executes | Total Cost |

|------------------------------------|-----------|------------------|------------|

| for all courses | 1 | n | n |

| if the course is the same as courseNumber | 1 | log n | log n |

| for each prerequisite of the course | 1 | 1 | 1 |

| print the prerequisite course information | 1 | n | n |

| Total Cost | | | n + log n |

| Runtime | | | O(n log n) |

Evaluation and Recommendation

Vector:

Advantages: Simple to implement and use, provides efficient indexing.

Disadvantages: Inefficient for insertion and deletion operations, O(n) for searching.

Runtime: O(n)

- \*\*Hash Table\*\*:

- Advantages: Fast search, insert, and delete operations (average O(1) time complexity).

- Disadvantages: Potential for high memory usage, performance depends on hash function.

- Runtime: O(n)

Binary Search Tree:

Advantages: Maintains sorted order, efficient search, insert, and delete operations (O(log n) for balanced trees).

Disadvantages: Can degrade to O(n) in unbalanced trees.

Runtime: O(n log n)

Based on the runtime analysis and the characteristics of each data structure, I recommend using the Hash Table for this project. It offers the best average-case performance for search, insert, and delete operations, which aligns well with the requirements to efficiently load, search, and print course information.