Paloma Rodriguez

CS300 : 6-2 Project One

SNHU

03/22/2023

Pseudocodes

**Vector :**

**For Creating Course Objects, Storage, Reading, Parsing, Checking Data:**

VECTOR courses

FUNCTION loadData(fileName)

OPEN fileName

FOR each line in file

check if line has at least two parameters

IF length of line >= 2

parse and store each parameter in variables

check if prerequisites provided exist in the file

IF all(prereq in courses for prereq)

create new course object and store in vector data structure

newCourse = Course(courseNum, courseId, prereqs)

courses.append(newCourse)

CLOSE file

**For Printing Course List in ABC orders:**

FUNCTION printCourseList()

SORT courses BY courseNum

FOR each course

PRINT course.courseNum, course.courseId

**For Searching the Data Structure and Printing Course Information and Prerequisites:**

FUNCTION printCourse(courseNum)

FOR each course

IF course.courseNum == courseNum

PRINT "Course Number:", course.courseNum

PRINT "Course Title:", course.courseId

IF course.prerequisites:

PRINT "Prerequisites:", ", ".join(course.prerequisites)

ELSE:

PRINT "Prerequisites: None"

RETURN

PRINT "Error"

**For Menu :**

FUNCTION main()

DISPLAY "Welcome"

WHILE True

DISPLAY "Please choose an option:"

DISPLAY "1. Load Data Structure"

DISPLAY "2. Print Course List"

DISPLAY "3. Print Course Information and Prerequisites"

DISPLAY "4. Exit"

READ userInput

IF userInput == 1

LOAD DATA "courseData.txt"

ELSE IF userInput == 2

PRINT COURSE LIST()

ELSE IF userInput == 3

READ courseNum

PRINT COURSE(courseNum)

ELSE IF userInput == 4

PRINT "Thank you"

BREAK

ELSE

PRINT "Error"

**Hash :**

**For Creating Course Objects, Storage, Reading, Parsing, Checking Data:**

HASH courses

FUNCTION loadData(fileName)

OPEN fileName

FOR each line in file

check if line has at least two parameters

IF length of line >= 2

parse and store each parameter in variables

check if prerequisites provided exist in the file

IF all(prereq in courses for prereq):

create new course object and store in hash table data structure

newCourse = Course(courseNum, courseID, prereqs)

CLOSE file

**For Printing Course List in ABC orders:**

FUNCTION printCourseList()

SORT keys of courses

FOR each key in sorted keys of courses

course = courses[key]

PRINT course.courseNum, course.courseId

**For Searching the Data Structure and Printing Course Information and Prerequisites:**

FUNCTION printCourse(courseNum)

IF courseNum in courses

course = courses[courseNum]

PRINT "Course Number:", course.courseNum

PRINT "Course Title:", course.courseId

IF course.prerequisites:

PRINT "Prerequisites:", ", ".join(course.prerequisites)

ELSE:

PRINT "Prerequisites: None"

ELSE

PRINT "Error"

**For Menu :**

FUNCTION main()

DISPLAY "Welcome"

WHILE True

DISPLAY "Please choose an option:"

DISPLAY "1. Load Data Structure"

DISPLAY "2. Print Course List"

DISPLAY "3. Print Course Information and Prerequisites"

DISPLAY "4. Exit"

READ userInput

IF userInput == 1

LOAD DATA "courseData.txt"

PRINT "Data loaded"

ELSE IF userInput == 2

PRINT COURSE LIST()

ELSE IF user\_input == 3

READ courseNum

PRINT COURSE(courseNum)

ELSE IF user\_input == 4

PRINT "Thank you"

BREAK

ELSE

PRINT "Error"

**Tree :**

**For Creating Course Objects, Storage, Reading, Parsing, Checking Data:**

FUNCTION addCourseToTree(root, course)

IF tree is empty, set course as root

IF root IS None

root = course

ELSE

Traverse tree to find correct position for course

currentNode = root

WHILE True

IF course number is less than current node, move to left child

IF course.courseNum < currentNode.courseNum

IF currentNode.leftChild is none

currentNode.leftChild = course

BREAK

ELSE

currentNode = currentNode.leftChild

ELSE

IF course number is greater than or equal to current node, move to right child

IF currentNode.rightChild is none

currentNode.rightChild = course

BREAK

ELSE

currentNode = currentNode.rightChild

**For Printing Course List in ABC orders:**

FUNCTION printCourseList(node)

Traverse tree in order and print course information

PRINT node.courseNum, node.courseId

**For Searching the Data Structure and Printing Course Information and Prerequisites:**

FUNCTION searchCourse(node, courseNum)

IF node is none OR node.courseNum == courseNum

RETURN node

IF course number is less than current node, move to left child

IF courseNum < node.courseNum

RETURN searchCourse(node.leftChild, courseNum)

IF course number is greater than current node, move to right child

ELSE

RETURN searchCourse(node.rightChild, courseNum)

**For Menu :**

FUNCTION main()

DISPLAY "Welcome"

WHILE True

DISPLAY "Please choose an option:"

DISPLAY "1. Load Data Structure"

DISPLAY "2. Print Course List"

DISPLAY "3. Print Course Information and Prerequisites"

DISPLAY "4. Exit"

READ userInput

IF userInput == 1

LOAD DATA "courseData.txt"

PRINT "Data loaded"

ELSE IF userInput == 2

PRINT COURSE LIST(root)

ELSE IF userInput == 3

READ courseNum

IF course IS NOT None

PRINT "Course Number:", course.courseNum

PRINT "Course Title:", course.courseId

ELSE IF userInput == 4

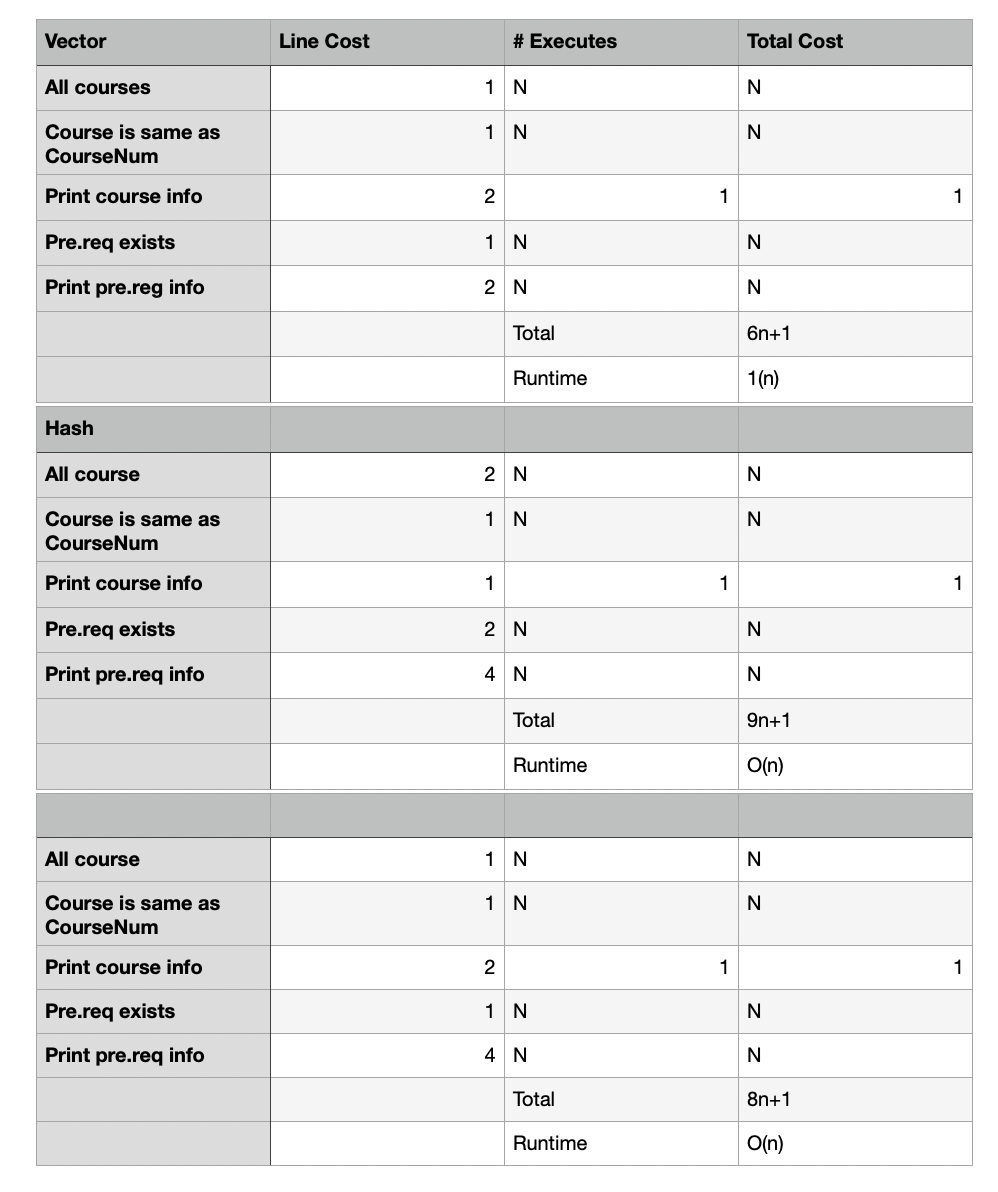
PRINT "Thank you"

BREAK

ELSE

PRINT "Error"

**Runtime Analysis Chart For Each Structure Type:**

****

**Advantages and disadvantages for each structure and recommendation :**

Vector:

Advantages: Easy to implement, low memory usage and best for small to medium size datasets

Disadvantages: Insertion and deletion of datasets can be slow and not suitable if frequent

Hash:

Advantages: Fast access to elements, best for large datasets and can handle insertions and deletions

Disadvantages: Memory usage can be high and can become complex

Tree:

Advantages: Great for searching, insertion and deletion, can handle large datasets

Disadvantages: Can become complex compared to hash and vector

Based on the runtime analysis, vectors would be best to use but there are many advantages and disadvantages to take into consideration when comparing data structures right for one’s project. Vector would be great to use if the dataset is on the smaller side and the focus is based on how fast one needs access to the elements. It is also great to implement and does not require much memory which can be an important factor if working in a limited enviroment.