Shaun Ryan

Professor Andujar

CS-300: DSA: Analysis and Design

10/10/2021

**Tree Pseudocode**

**void Menu()**

INITIALIZE choice to 0

WHILE choice does not equal 4

OUTPUT menu and prompt to screen

SET choice to user input

IF choice equals 1

PROMPT for file name

SET userInput to user input

CALL LoadTree(userInput)

ELSE IF choice equals 2

CALL PrintCourseList()

ELSE IF choice equals 3

PROMPT for course name

SET userInput to user input

CALL PrintCourseInformation(userInput)

OUTPUT “Goodbye” to the screen

**void LoadTree(string userInput)**

INITIALIZE holderString

INITIALIZE prerequisiteVector, courseVector, infileVector, holderVector

INITIALIZE inFileStream

OPEN dataFile userInput

LOOP until end of datafile

SET holderString to dataFile line

INITIALIZE Course newCourse

LOOP until end of holderString

SET Course members to holderString elements

IF newCourse number or name are empty

PRINT error “Not enough parameters for course”

EXIT

CALL insert(newCourse)

FOR all prerequisites in tree

FOR all course numbers in tree

CHECK that prerequisite is in courses

PRINT “Course list loaded successfully”

CLOSE datafile

RETURN

**void Insert(Node current, Course newCourse)**

CHECK if tree root is empty

SET root to newCourse

SET left and right to null

CHECK if course number is less than the current node

TRAVERSE left

CHECK if node is empty

ADD new node with newCourse as a parameter

ELSE

CALL Insert(current->left, newCourse) recursively

CHECK if course number is grester than the current node

TRAVERSE right

CHECK if node is empty

ADD new node with newCourse as a parameter

ELSE

CALL Insert(current->right, newCourse) recursively

**void printCourseInformation(courseNumber)**

INITIALIZE pointer current

SET current to the root node

LOOP until current equals null

CHECK if courseNumber is equal to current’s course number

OUTPUT current’s course information to the screen

RETURN

CHECK if courseNumber is less than current’s course number

TRAVERSE left

CHECK if courseNumber is greater than current’s course number

TRAVERSE right

OUTPUT “Course not found”

RETURN

Runtime Analysis

| **LoadTree()** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **INITIALIZE variables** | 6 | 1 | 6 |
| **OPEN dataFile** | 1 | 1 | 1 |
| **LOOP until end of dataFile** | 1 | n | n |
| **SET holderString to dataFile line** | 1 | n | n |
| **INITIALIZE newCourse** | 1 | n | n |
| **LOOP until end of holder string** | 1 | 4n | 4n |
| **SET newCourse members to holder sting** | 1 | 4n | 4n |
| **IF newCourse missing members** | 1 | n | n |
| **PRINT error** | 1 | 0 | 0 |
| **CALL insert** | 7n + 7 | n | 7n^2+7n |
| **FOR all prerequisites in tree** | 1 | n | n |
| **FOR all course numbers in tree** | 1 | n^2 | n^2 |
| **CHECK that prerequisite is in courses** | 1 | n^2 | n^2 |
| **PRINT completion message** | 1 | 1 | 1 |
| **CLOSE datafile** | 1 | 1 | 1 |
| **Total Cost** | | | 9n^2 + 20n + 9 |
| **Runtime** | | | O(n^2) |

| **Insert()** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **CHECK if tree root is empty** | 1 | 1 | 1 |
| **SET root to newCourse** | 1 | 1 | 1 |
| **SET left and right to null** | 1 | 1 | 1 |
| **CHECK if course number is less than current node** | 1 | 1 | 1 |
| **TRAVERSE left** | 1 | 1 | 1 |
| **CHECK if node is empty** | 1 | 1 | 1 |
| **ADD new node** | 1 | 0 | 0 |
| **ELSE** | 1 | 1 | 1 |
| **CALL Insert()** | 7 | n | 7n |
| **CHECK if course is greater than current node** | 1 | 0 | 0 |
| **TRAVERSE right** | 1 | 0 | 0 |
| **CHECK if node is empty** | 1 | 0 | 0 |
| **ADD new node** | 1 | 0 | 0 |
| **ELSE** | 1 | 0 | 0 |
| **CALL Insert()** | 1 | 0 | 0 |
| **Total Cost** | | | 7n + 7 |
| **Runtime** | | | O(n) |