Joshua Hale

July 11, 2024

**CS300**

**5-3 Project One Milestone Three: Tree Data Structure Pseudocode**



**Pseudocode for Loading Data into a Binary Search Tree**

// Define a Course class

class Course:

string courseNumber

string name

List<string> prerequisites

// Constructor to initialize the course object

Course(string courseNumber, string name, List<string> prerequisites):

this.courseNumber = courseNumber

this.name = name

this.prerequisites = prerequisites

// Define a TreeNode class for the binary search tree

class TreeNode:

Course course

TreeNode left

TreeNode right

// Constructor to initialize the tree node

TreeNode(Course course):

this.course = course

this.left = null

this.right = null

// Define a BinarySearchTree class

class BinarySearchTree:

TreeNode root

// Constructor to initialize the binary search tree

BinarySearchTree():

this.root = null

// Method to insert a course into the tree

void insert(Course course):

if root == null:

root = new TreeNode(course)

else:

insertRec(root, course)

// Recursive method to insert a course into the tree

void insertRec(TreeNode node, Course course):

if course.courseNumber < node.course.courseNumber:

if node.left == null:

node.left = new TreeNode(course)

else:

insertRec(node.left, course)

else:

if node.right == null:

node.right = new TreeNode(course)

else:

insertRec(node.right, course)

// Function to load data from file and store it in the binary search tree

void loadData(string filename):

BinarySearchTree bst = new BinarySearchTree()

file = open(filename, "r")

lines = file.readlines()

// Loop through each line in the file

for line in lines:

data = line.split(",")

courseNumber = data[0]

name = data[1]

prerequisites = data[2:]

// Check if the line has at least two parameters

if len(data) < 2:

print("Error: Invalid format. Each line must have at least a course number and name.")

continue

// Validate prerequisites

for prereq in prerequisites:

if not existsInFile(prereq, lines):

print(f"Error: Prerequisite {prereq} does not exist as a course in the file.")

continue

// Create a new course object

Course course = new Course(courseNumber, name, prerequisites)

bst.insert(course)

file.close()

return bst

// Function to check if a course exists in the file

bool existsInFile(string courseNumber, List<string> lines):

for line in lines:

data = line.split(",")

if data[0] == courseNumber:

return true

return false

// Function to print course information and prerequisites

void printCourseInformation(BinarySearchTree bst):

printInOrder(bst.root)

// Recursive function to print the courses in order

void printInOrder(TreeNode node):

if node != null:

printInOrder(node.left)

print(f"Course Number: {node.course.courseNumber}, Name: {node.course.name}, Prerequisites: {', '.join(node.course.prerequisites)}")

printInOrder(node.right)

**Explanation**

1. **Loading Data:**
   * The pseudocode reads data from a file line by line.
   * Each line is split into tokens, where the first token is the course number, the second is the course name, and the rest are prerequisites.
   * The pseudocode checks for file format errors and validates the prerequisites.
   * It creates a course object for each valid line and inserts it into the binary search tree.
2. **Course Object:**
   * A Course class is defined to hold the course number, name, and prerequisites.
   * A TreeNode class is defined to represent each node in the binary search tree.
   * A BinarySearchTree class is defined with methods to insert courses and print the tree in order.
3. **Printing Course Information:**
   * A function printCourseInformation traverses the binary search tree and prints each course's information and prerequisites.

***Image Reference:***

<https://www.geeksforgeeks.org/types-of-trees-in-data-structures/>