// Binary Search Tree pseudocode

BinarySearchTree ParseCourseData(string filepath) {

create BinarySearchTree courseData

if input file missing

throw FileNotFound exception

open input file at filepath

for each line l in input file

split line along commas into array

if array length less than 2

continue to next line

for each prerequisite

loop through input file lines

take first comma-delimited substring

if course ID matches prerequisite

success, continue to prerequisite loop

if no match is found

continue to next line

Course newCourse = Course(number, title, prerequisites[])

courseData.insert(newCourse)

}

BinarySearchTree() {

Node root = nullptr;

}

Node(Course aCourse) {

course = aCourse

Node left = nullptr;

Node right = nullptr;

}

Course(string courseNumber, string courseTitle, char\* coursePrerequisites[]) {

number = courseNumber

title = courseTitle

prerequisites = coursePrerequisites

}

int numPrerequisiteCourses(BinarySearchTree courses, Course c) {

totalPrerequisites = prerequisites of course c

for each prerequisite p in totalPrerequisites

totalPrerequisites += numPrerequisites(courses, p)

return totalPrerequisites

}

void printSampleSchedule(BinarySearchTree courses) {

get last element in courses

print course information for final element

}

void printCourseInformation(BinarySearchTree courses, String courseNumber) {

currentNode = courses.search(courseNumber);

if currentNode == nullptr {

return

} else if currentNode.number matches courseNumber {

print course number, title, and prerequisites

return

}

}

ParseCourseData Runtime Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Line | Line Cost | # Times Execute | Total Cost |
| For each line l in input file | 1 | n | n |
| Loop through input file lines | 1 | n^2 | n^2 |
| courseData.insert(newCourse) | log n | n | n log n |
| Total cost: | n + n^2 + n log n | Runtime: | O(n ^ 2) |

ParseCourseData Memory Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Line | Line Cost | # Times Execute | Total Cost |
| Create Vector<Course> CourseData | n | 1 | n |
| Split line along commas into array | 1 | m | m |
| Course newCourse = Course(number, title, prerequisites[]) | 1 | N | n |
| Total cost: | 2n + m | Memory: | O(n + m) |

where M is the maximum length of a line in the file