// Hash pseudocode

HashTable ParseCourseData(string filepath) {

create HashTable 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

int newKey = hash(courseNumber)

Node newNode = new Node(Course(), newKey)

courseData.insert(key, newNode)

}

HashTable() {

vector<Node> nodes

int tableSize = DEFAULT\_SIZE

}

Node(Course aCourse, int aKey) {

course = aCourse

key = aKey

nextNode = nullptr;

}

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

number = courseNumber

title = courseTitle

prerequisites = coursePrerequisites

}

int numPrerequisiteCourses(HashTable courses, Course c) {

totalPrerequisites = prerequisites of course c

for each prerequisite p in totalPrerequisites

totalPrerequisites += numPrerequisites(courses, p)

return totalPrerequisites

}

void printSampleSchedule(HashTable courses) {

get last element in courses

print course information for final element

}

void printCourseInformation(HashTable courses, String courseNumber) {

key = hash(courseNumber)

currentNode = get courses(key)

if currentNode.number matches courseNumber {

print course number, title, and prerequisites

return

}

else if currentNode is unused or empty {

return

} else {

Iterate through node chain

if currentNode.number matches courseNumber {

print course number, title, and prerequisites

return

}

set currentNode = currentNode’s next node

}

}

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(key, newNode) | 1 | n^2 | n^2 |
| Total cost: | n + 2n^2 | Runtime: | O(n ^ 2) |

ParseCourseData Memory Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Line | Line Cost | # Times Execute | Total Cost |
| Create HashTable CourseData | n | 1 | n |
| Split line along commas into array | 1 | m | m |
| Node newNode = new Node(Course(), newKey) | 1 | n |  |
| Total cost: | n + m | Memory: | O(n + m) |

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