# Greg Monti

# CS 300 Pseudocode Document

// Vector pseudocode

**//opening file, reading data, creating course**

**Open file courseData**

**For each line**

**courseData = line**

**course = newCourse**

int numPrerequisiteCourses(Vector<Course> courses, Course c) {

totalPrerequisites = prerequisites of course c

for each prerequisite p in totalPrerequisites

add prerequisites of p to totalPrerequisites

print number of totalPrerequisites

}

void printSampleSchedule(Vector<Course> courses) {

**sort courseList**

**For course in course list**

**Print courseList**

**Print courseTitle**

**print coursePrereqs**

}

void printCourseInformation(Vector<Course> courses, String courseNumber) {

**for all courses**

**if the course is the same as courseNumber**

**print out the course information**

**for each prerequisite of the course**

**print the prerequisite course information**

}

// Hashtable pseudocode

**//opening file, reading data, creating course**

**Open file courseData**

**For each line**

**courseData = line**

**courseTable.inset = newCourse**

int numPrerequisiteCourses(Hashtable<Course> courses) {

**totalPrerequisites = Hashtable**

**sort courseList**

**for each prerequisite in totalPrerequisites, add prerequisite in Hashtable to totalPrerequisites**

**print number of totalPrerequisites**

}

void printSampleSchedule(Hashtable<Course> courses) {

**print main course name**

**if value has prerequisites**

**print prerequisites**

}

void printCourseInformation(Hashtable<Course> courses, String courseNumber) {

**All courses**

**If value is same as courseNumber**

**Print course information**

**For prerequisites contained in the Hashtable**

**Print the prerequisite course information**

}

// Tree pseudocode

**//opening file, reading data, creating course**

**Open file courseData**

**For each line**

**courseData = line**

**course = newCourse**

**node = new treeNode**

int numPrerequisiteCourses(Tree<Course> courses) {

**Creates nodes in totalPrerequisites for each value of prerequisite**

}

void printSampleSchedule(Tree<Course> courses) {

**Applies to all nodes**

**Print course name**

**If in prerequisite node**

**Print node as a prerequisite**

**if in other node**

**print course information**

}

void printCourseInformation(Tree<Course> courses, String courseNumber) {

**Applies to all nodes**

**If int is same as courseNumber**

**Print out node information**

**If left node**

**Print information as prerequisite course**

**If right node**

**Print information as prerequisite course**

}

## Example Runtime Analysis

When you are ready to begin analyzing the runtime for the data structures that you have created pseudocode for, use the chart below to support your work. This example is for printing course information when using the vector data structure. As a reminder, this is the same pairing that was bolded in the pseudocode from the first part of this document.

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 1 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 1 | n | n |
| **Total Cost** | | | 4n + 1 |
| **Runtime** | | | O(n) |

| **Code Hash Table** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **If value is same as courseNumber** | 1 | n | n |
| **print out the course information** | 1 | 1 | 1 |
| **For prerequisites contained in the Hashtable** | 1 | n | n |
| **Print the prerequisite course information** | 1 | n | n |
| **Total Cost** | | | 4n + 1 |
| **Runtime** | | | O(n) |
| **Code Tree** | **Line Cost** | **# Times Executes** | **Total Cost** |
| **for all courses** | 1 | n | n |
| **If int is same as courseNumber** | 1 | n | n |
| **print out the course information** | 1 | 1 | 1 |
| **If left node** | 1 | n | n |
| **Print information as prerequisite course** | 1 | 1 | 1 |
| **If right node** | 1 | n | n |
| **Print information as prerequisite course** | 1 | n | n |
| **Total Cost** | | | 5n + 2 |
| **Runtime** | | | O(n) |

**Recommendation**

There are many ways to store and access large amounts of data such as what we are looking at with courses. There will obviously be a lot of data stored within these structures so we need an efficient and timely way to provide this information to students. That being said, a tree would accomplish this. Because we want this to run as efficiently as possible, using a tree data structure will make sure the runtime is as quick as possible. Being able to search through the tree at fast rates, it will allow us to view, add, or remove from the tree with ease.