**Adam Conger**

**SNHU CS 300**

**Project 1**

Pseudocode for Prerequisites

void fileInput(Vector<Course> courses) {

**load file from file path**

**for each line**

**read line**

**create new courseObject**

}

void courseObject(Vector<Course> courses, String courseNumber, String courseTitle, String prerequisite) {

**start at index 0**

**for line**

**store courseNumber**

**store courseTitle**

**if there is another value**

**if value exists as courseNumber**

**store prerequisite**

}

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**

}

void fileInput(HashTable<Course> courses) {

**load file from file path**

**create hash key**

**for each line**

**read line**

**create new courseObject**

**assign key to courseObject**

}

void courseObject(HashTable<Course> courses, String courseNumber, String courseTitle, String prerequisite) {

**create hash key**

**for each line**

**store courseNumber**

**store courseTitle**

**if there is another value**

**if value exists as courseNumber**

**store prerequisite**

}

void printCourseInformation(HashTable<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**

}

void fileInput(BinaryTree<Course> courses) {

**load file from file path**

**for each line**

**read line**

**create new courseObject**

**if courseObject is less than 0**

**assign left to courseObject**

**else**

**assign right to courseObject**

}

void courseObject(BinaryTree<Course> courses, String courseNumber, String courseTitle, String prerequisite) {

**for each line**

**store courseNumber**

**store courseTitle**

**if there is another value**

**if value exists as courseNumber**

**store prerequisite**

}

void printCourseInformation(BinaryTree<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**

}

Pseudocode for Menu

void menu() {

**load cases 1, 2, 3, 9**

**output user options**

**case 1:**

**fileInput(courses)**

**case 2:**

**printCourses(courses)**

**case 3:**

**printCourseInformation(courseNumber)**

**case 9:**

**exit**

}

Pseudocode for Course List

void printCourses(courses) {

**load data structure**

**for all in data structure**

**read element**

**if element matches alphaKey**

**advance alphaKey**

**return element**

}

Run-Time Analysis

| **fileInput Vector** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Load file from file path** | 1 | 1 | 1 |
| **For each file** | 1 | n | n |
| **Read line** | 1 | n | n |
| **Create new courseObject** | 1 | n | n |
|  |  | **Total Cost** | 3n + 1 |
| **Runtime** | | | O(n) |

| **courseObject** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **For line(start at index 0)** | 1 | n | n |
| **store courseNumber** | 1 | 1 | 1 |
| **store courseTitle** | 1 | 1 | 1 |
| **if there is another value** | 1 | n | n |
| **if value exists as courseNumber** | 1 | n | n |
| **Store prerequisite** | 1 | 1 | 1 |
| **Total Cost** | | | 3n + 3 |
| **Runtime** | | | O(n) |

| **printCourseInformation** | **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) |

| **fileInput Hash Table** | **Line Cost** | **# Times Executes** | **Total Cost** | |
| --- | --- | --- | --- | --- |
| **Load file from file path** | 1 | 1 | 1 | |
| **Create hash key** | 1 | 1 | 1 | |
| **For each line** | 1 | n | n | |
| **Read line** | 1 | n | n | |
| **Create new courseObject** | 1 | n | n | |
| **Assign key to courseObject** | 1 | n | n | |
| **Total Cost** | | | 4n+2 | |
| **Runtime** | | | O(n) | |
|  | | |  | |
| **courseObject** | **Line Cost** | **# Times Executes** | **Total Cost** |
| **Create hash key** | 1 | 1 | 1 |
| **For line(start at index 0)** | 1 | n | n |
| **store courseNumber** | 1 | 1 | 1 |
| **store courseTitle** | 1 | 1 | 1 |
| **if there is another value** | 1 | n | n |
| **if value exists as courseNumber** | 1 | n | n |
| **Store prerequisite** | 1 | 1 | 1 |
| **Total Cost** | | | 3n + 4 |
| **Runtime** | | | O(n) |

| **printCourseInformation** | **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) |

| **fileInput Binary Search Tree** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Load file from file path** | 1 | 1 | 1 |
| **For each file** | 1 | n | n |
| **Read line** | 1 | 1 | 1 |
| **Create new courseObject** | 1 | 1 | 1 |
| **If courseObject is less than 0** | 1 | n | n |
| **Assign left to courseObject** | 1 | 1 | 1 |
| **Otherwise assign right to courseObject** | 1 | 1 | 1 |
|  |  | **Total Cost** | 2n + 5 |
| **Runtime** | | | O(n) |

| **courseObject** | **Line Cost** | **# Times Executes** | **Total Cost** | |
| --- | --- | --- | --- | --- |
| **For line(start at index 0)** | **1** | **n** | **n** | |
| **store courseNumber** | **1** | **1** | **1** | |
| **store courseTitle** | **1** | **1** | **1** | |
| **if there is another value** | **1** | **n** | **n** | |
| **if value exists as courseNumber** | **1** | **n** | **n** | |
| **Store prerequisite** | **1** | **1** | **1** | |
| **Total Cost** | | | | 3n + 3 | |
| **Runtime** | | | | O(n) | |

| **printCourseInformation** | **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) |

Advantages and Disadvantages

The vector method has the fastest runtime. Using a hash table may take up more memory and may not be necessary for a course database. In the long-term, it may make sense to implement search trees, and each course department would be its own tree, searching on the course number within that department (that way there’s not a collision with MAT1000 and CS1000, for instance).

Recommendations

I think for the scope of the data we have, using a vector or linked list will make sense for the short and mid-term usage. In the long term, as more departments are wound into the project, it could be transitioned to a binary search tree structure for each department’s course offerings.