# CS 300 Pseudocode Document

**Vector Pseudocode**

//Open, Read//

Call to open file

If return -1

No file

Else

Read lines

If no values

Return error

Else

Read

Close

//Insert//

Initialize vector<int>courseInfo()

Loop

While not end

For each line

First and second values

Add to vector

Third value

Pushback to new line

//Print//

Request user input

Loop vector

If input courseNumber

Print course information

If prerequisites exist

Print prerequisite information

Hash Table Pseudocode

//Open, Read//

Call to open file

If return -1

No file

Else

Read lines

If less than two values

Error

Else

Read

Close

//Insert//

Initialize nodes

Create hash table

Create insert method

Assign key to current node

Search for node with key

If no node found

Assign to key

Else

Find next open node

Loop through file

While not end

Values 1 and 2

Values 3+

Add to current

//Print//

Request user input

Assign as key

If key found

Print course information

If prerequisites found

Print prerequisites information

**Tree Data Structure Pseudocode**

//Open, Read//

Call to open file

If return = -1

No file

Else

Read lines

Close

//Insert//

Initialize nodes

Create binary tree

Create root

Create insert method

If root = null

Assign current to root

Else

If course # less than root

Add left branch

If course # greater than root

Add right branch

Loop through file

While not end of file

1st and 2nd value

Add name & ID

//Insert//

Request user input

If root not null

Traverse left

If found, print info and prerequisites

Else

Traverse right

If found, print course info and prerequisites

**Menu Pseudocode**

Create while loop

Request user input

While user input not 4

Output menu options

1. loadCourses

Load data into structure

1. printCourseList

Print ordered list of courses

1. printCourse

Print course name and prerequisites

1. exit

**Print Alphanumeric Pseudocode**

**Vector**

Create alphanumeric sort method

Set low

First (CSCI100)

Set high

Last (MATH201)

Print

Print alphanumeric sorted list

**Hash Table**

Convert hash table keys to list method

Sort list alphanumerically

Low (CSCI100)

High (MATH201)

Print alphanumeric list

**Tree**

Traverse tree in order

During traverse

Print node course information

**Evaluation**

| **Vector** | **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** | 2 | n | n |
| **Total Cost** | | | 5n + 1 |
| **Runtime** | | | O(n) |

| **Hash Table** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 2 | 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** | 2 | n | n |
| **print the prerequisite course information** | 3 | n | n |
| **Total Cost** | | | 8n + 1 |
| **Runtime** | | | O(n) |

| **Tree** | **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** | 2 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 4 | n | n |
| **Total Cost** | | | 8n + 1 |
| **Runtime** | | | O(n) |

Each data structure that we have learned about in this course has its own pros and cons. For example, a vector generally has the advantage of being the fastest method, despite the con of having to search for each specific course separately. With the evaluation, I found vector to be the lowest cost at 5n+1.

A hash table is able to search the entire list with the use of a key. With said key, the search for an item, in this case a course, can be done quickly. However, hash tables are generally slower to create as a method as compared to the other two. In addition, a table itself cannot be sorted. When sorting alphanumerically, you need to convert the table into a list via the keys. That list can then be sorted. With that in mind, the cost is quite high at 8n+1.

Trees have a create advantage in terms of what is being asked for this assignment, being that sorting a tree is very quick. The natural structure of a binary search tree is one of organization, with lower values taking the left position and higher values taking the right. While quick to sort, the cost is still higher than a vector at 8n+1.

With all of this in mind, my recommendation for method for this assignment is the vector method. With the low cost and speed, it seems like a pretty each choice.