Pseudocode:

Import csv files containing courses

Struct Course{

courseNumber

courseName

preReq

fileOpen(){

while file can open

if end of file != true

read file lines and check if there are at least 2 parameters on line

close file

else if end of file = true

close file and print end of file reached}

createCourse(Vector<course> courses, String courseNumber, string preReq){

Create HashTable{

Initialize table size

Create key{

Return key % tableSize}

open csv and loop file

while end of file != true

each line = first and second value in hash table

add first value as course name and second value as course number

if there is more than 3 parameters

loop until no more parameters are present

each parameter after the first = preReq

}

searchAndPrintCourse(){

Create tempTable

Loop through tempTable

IF input is found in tempTable

Set tempCourse = course and return course info and prereqs

} Struct Course{

courseNumber

courseName

preReq

fileOpen(){

while file can open

if end of file != true

read file lines and check if there are at least 2 parameters on line

close file

else if end of file = true

close file and print end of file reached}

createCourse(Vector<course> courses, String courseNumber, string preReq){

Create binaryTree{

Create root = NULL;

Insert node{

If root = null

Course = root

Else if course < root

Add left

Else if course > root

Add right

open csv and loop file

while end of file != true

each line = first and second value in BST

add first value as course name and second value as course number

if there is more than 3 parameters

loop until no more parameters are present

each parameter after the first = preReq

}

searchAndPrintCourse(){

Create tempNode

Loop through root

IF input is found in root

Set tempCourse = course and return course info and prereqs

}

Struct Course{

courseNumber

courseName

preReq

fileOpen(){

while file can open

if end of file != true

read file lines and check if there are at least 2 parameters on line

close file

else if end of file = true

close file and print end of file reached}

createCourse(Vector<course> courses, String courseNumber, string preReq){

open csv and loop file

while end of file != true

each line = first and second value in vector

add first value as course name and second value as course number

if there is more than 3 parameters

loop until no more parameters are present

each parameter after the first = preReq

}

void searchCourse(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**

}}

Menu Pseudocode:

Int choice = 0

While true,

Switch choice:

Case 1: Load course data into data structure,

Case 2: Print alphanumeric ordered list of courses in CS dept.

Case 3: Print course titleand prereqs

Case 4: exit

Else

Print “wong choice!”

Sorted list Pseudocode:

Create sortedList{

Declare

Int partition(vector, begin, end)

Int low = begin

Int high = end

Int pivot = begin + end

Bool done = false

While not done{

While low is less than pivot, increase low by 1;

While pivot is less than high, decrease high by 1;

If low is >= high, set done to true;

…

Main()

Load csvPath and set default location.

Define vector for bids and clock

Display menu

Run time analysis:

|  |  |  |
| --- | --- | --- |
| Data Types | Total cost | Run time |
| Vector | 6n + 1 | 1(n) |
| Hash Table | 9n + 1 | O(n) |
| Tree | 8n + 1 | O(n) |

Each structure used in this pseudocode have their advantages, such as the vector being simple to use and faster, the hash table being superior at holding large data sets, and the binary tree being over all the most efficient at searching and retrieving data. The disadvantages of each of the structures, on the other hand, dictates which one should be used in this circumstance of sorting courses as well as their prerequisites. The major disadvantage of the vector is the fact that it is quite substantially slower than both the hash table and the binary tree at searching. The hash table falls short on the memory requirements for holding the actual data. Finally, the binary tree would take the longest to sort.

Based upon these advantages and disadvantages, I believe that using the binary tree in the long run would be the best thing to do. Despite it taking the longest time to initially sort the data, accessing the data and editing it proves to be superior to the vector and hash table. The Binary tree is an easily maintainable and updatable data structure that would easily hold the course data and be searched.