Project One

By: Trayven Nixon

Sathish Gopalakrishnan

CS-300-13175-M01 DSA: Analysis and Design

February 18, 2025

**1.a Pseudocode for Vector**

Include libraries and headers

Declare string filePath

Set int userChoice = 0

Declare string userSearch

Define a structure to hold Course information

courseNumber //course ID number

courseTitle //name of course

preReqs //prerequisites for course

Function fileCheck(filePath) // checks if file is readable

If file opens

while file can be opened and there are at least two values per line

iterate through file begging to end

If there is a third value

If it matches a first value

Return True

Continue loop

Else

Return False

End loop

Else

Do nothing

Else

Return False

Function loadCSV(filePath, CourseList) //open .csv file and add values from file to Course structure

Call fileCheck(filePath)

If check is true

Open file path

For each line in .csv file

Split line at “, or | ”

Append courseList with values.

courseNumber = first value

courseTitle = second value

while column still has values

increment preReqs adding remaining values

close file

Else

Print file is not compatible.

Close file

Function getUserSearchID()

Declare String tempID

Print “What course would you like to search for?”

Print “Please enter course ID number: “

tempID = user input

if user input is equal to 6 characters that are not symbols

courseNumber = tempID

print “searching”

else

print “You have entered an invalid course ID”

Function searchCourse(CourseList, 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

Function userPath(user input)

Print “Please input the file path you wish to use”

filePath = User input

**1.b Pseudocode for Hash Table**

Include libraries and headers

Declare string filePath

Set int userChoice = 0

Declare string userSearch

Define a structure to hold Course information

courseNumber //course ID number

courseTitle //name of course

preReqs //prerequisites for course

Define class HashTable node

Course is course

initialize key

initialize Node

define constructor Node

key equals max

next till null

Initialize with course

Node(Course, aCourse and Node

course = Course

Initialize with a course and a key

Node() : Node(aBid)

Key equals aKey

Initialize table side = TableSize = DEFAULT\_SIZE of the table

Initialize hash(key)

Function HashTable::HashTable() //Create Default constructor

Initialize with default table size

Function HashTable::Hashtable(size)

Set table size to default table size

Set resizing parameters

Function HashTable::hash() //sets type of hash to use

Return a key using the % hash type

Function fileCheck(filePath) // checks if file is readable

If file opens

while file can be opened and there are at least two values per line

iterate through file begging to end

If there is a third value

If it matches a first value

Return True

Continue loop

Else

Return False

End loop

Else

Do nothing

Else

Return False

Function loadCSV(filePath, Course) //open .csv file and add values from file to Course structure

Create key for bid // use atoi and c\_string

Call fileCheck(filePath)

If check is true

Open file path

For each line in .csv file

Split line at “, or | ”

Append Course with values.

courseNumber = first value

courseTitle = second value

while column still has values

increment preReqs adding remaining values

Insert course to Course hash table

close file

Else

Print to screen, file is not compatible.

Close file

Function getUserSearchID()

Declare String tempID

Print “What course would you like to search for?”

Print “Please enter course ID number: “

tempID = user input

if user input is equal to 6 characters that are not symbols

courseNumber = tempID

print “searching”

else

print “You have entered an invalid course ID”

Function searchCourse(courseNumber)

Define node

Create key for given bid using hash

Get node using key

Loop over list to find bid

If found

Print out the course information

For each prerequisite of the course

Print the prerequisite course information

Else

Print no match found

Function userPath(user input)

Print “Please input the file path you wish to use”

filePath = User input

**1.c Pseudocode for Binary Tree**

Include libraries and headers

Declare string filePath

Set int userChoice = 0

Declare string userSearch

Define a structure to hold Course information

courseId //course ID number

courseTitle //name of course

preReqs //prerequisites for course

Define class BinaryTree

Node structure

courseId

right pointer

left pointer

Function fileCheck(filePath) // checks if file is readable

If file opens

while file can be opened and there are at least two values per line

iterate through file begging to end

If there is a third value and if it matches a first value

Return True

Continue loop

Otherwise return false and end loop

Return False

Function loadCSV(filePath, Course) //open .csv file and add values from file to Course structure

Call fileCheck(filePath)

If check is true

Open file path

For each line in .csv file

Split line at “, or | ”

Append Course with values.

courseNumber = first value

courseTitle = second value

while column still has values

increment preReqs adding remaining values

Insert course to Course binary tree

close file

Else

Print to screen, file is not compatible.

Close file

Function getUserSearchID()

Declare String tempID

Print “What course would you like to search for?”

Print “Please enter course ID number: “

tempID = user input

if user input is equal to 6 characters that are not symbols

courseId= tempID

print “searching”

else

print “You have entered an invalid course ID”

Function searchCourse(courseId)

Create tempCourse node

Loop through tree to find matching courseId

If found

Print out the course information

For each prerequisite of the course

Print the prerequisite course information

Else

Print no match found

Function userPath(user input)

Print “Please input the file path you wish to use”

filePath = User input

**2. Menu Pseudocode**

Function menu(userChoice)

While userChoice does not equal 9

Print “Menu:”

Print “1. Load files”

Print ”2. Search courses”

Print “3. Print Courses in alphabetical order”

Print “9. Exit”

userChoice = user input

if userChoice = 1

call userPath()

call LoadCSV()

break

if userChoice = 2

call alphaPrintCourses()

if userChoice = 3

call userSearchID()

call searchCourse()

break

if userChoice = 9

end program

**3. Pseudocode to print in Alphanumeric order**

sortCoursesVector()

starting from 0 and for the length of the course -1

if the courseId is greater than the course before it, swap course, continue linearly

Then for each courseId in course, print course and prerequisite to screen

sortCoursesHashtable()

Sort by key starting at 0

If there is more than one element for a key

Check which is lowest and swap, continue for every element for that key

Then for each courseId in course, print course and prerequisite to screen

sortCoursesBinaryTree()

while node is not null

traverse the left side of the tree

print the current node

traverse the right side of the tree

**4. Evaluate run time**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Vector** | **Hash Table** | **Binary Tree** |
| **Loading Courses** | O(n) | O(n) | O(n) |
| **Search** | O(n) | O(1) | O(log n)  *Worst case O(n)* |
| **Sort / Print** | O(n log n)  *After sorting print time would be O(n)* | O(log n)  *Worst case O(n)* | O(n) |

**5. Advantages and disadvantages of each structure**

* **Vector**
  + Unsorted vectors are fast to load using the push back method but will be unable to use binary search. The disadvantage is that you would need to do a linear search for the sorting and search functions which takes more time than the other 2 structures.
* **Hash Table**
  + Hash tables are efficient for inserting, searching, and printing with a O(1) time complexity. Hash tables use memory efficiently using dynamic resizing. Hash tables are not sorting and depending on collisions can be difficult to sort or search.
* **Binary Tree**
  + Binary trees are already mostly sorted, which makes traversal for searching, sorting or printing the same or quicker than a vector. It has the same speed as the hash table when it comes to sorting and print but depending on the balance of the tree can fall short to the hash table. Binary trees can also utilize more memory because they require pointers from parents to children.

**6. Recommendation**

Because the primary role of this program is to look up courses, I recommend using a hash table. Hash tables excel when requiring fast access to elements. Also, when using dynamic resizing, it can minimize performance issues due to collisions.