**6-2 Project One**

Christopher King

Southern New Hampshire University

CS-300 DSA: Analysis and Design

Michael Susalla

October 8, 2023

**6-2 Project One**

**Vector Data Structure Pseudocode**

**Pseudocode to read data from a file**

Use fstream to be able to open file

Call to open file

Check if file opened

If return is Null

File not found

Else (file found)

While not end of file

Set valueCount to 0

Get line from fstream

Get valueCount from line up to “,” until end of line

If valueCount is less than 2

Display error

Else if valueCount is greater than 2

If third or greater parameter is present elsewere as parameter one

Continue

Else

Return error

Close File

**Pseudocode to Create and store course objects**

Create struct Course

Create identifiers: courseId, couresName, prerequisite

Initialize Course Vector

Open file for parsing

WHILE it is not the end of the file

FOR each line in file

FOR first and second value (courseId, courseName)

USE pushback to add values to vector

IF more than 2 values (prerequisites)

USE pushback to add values to vector

**Pseudocode to search data structures and print course information**

ASK for input

WHILE it is not the end of the file

IF course found

PRINT course information

FOR each prerequisite

PRINT prerequisite

ELSE

PRINT “Course not found”

**Hash Table Data Structure Pseudocode**

**Pseudocode to read data from a file**

Use fstream to be able to open file

Call to open file

Check if file opened

If return is Null

File not found

Else (file found)

While not end of file

Set valueCount to 0

Get line from fstream

Get valueCount from line up to “,” until end of line

If valueCount is less than 2

Display error

Else if valueCount is greater than 2

If third or greater parameter is present elsewere as parameter one

Continue

Else

Return error

Close File

**Pseudocode to create course objects and store in appropriate data structures**

Create HashTable Class

Create insert method to insert items into HashTable

Create key for course

Search for node with key

If no entry found

Assign current node to key

Else if current node is taken

Assign old node key to UNIT\_MAX

Set current node to key

Set old not to course

Set old node next to null pointer

Else fine next open node

Add new node to end

Create loadCourses method

While not end of file

For each line in file

For each parameter in line

Create temporary item to hold parameter values

Call insert method to insert into HashTable

**Pseudocode to print out course information and prerequisites**

ASK for input

WHILE it is not the end of the file

IF course found

PRINT course information

FOR each prerequisite

PRINT prerequisite

ELSE

PRINT “Course not found”

**Tree Data Structure Pseudocode**

**Pseudocode that opens the file, reads the data from the file, parses each line, and checks for file format errors:**

Start

Create left node variable

Create right node variable

Create sting course name variable

Create int course number variable

Open file

While file is open

For each line

Parse each line

If root is not equal to null

If prerequisite found

Add prerequisite to right node

If course parameter count is less than two

Display error

Exit

Else

Add course name, number, and prerequisite

Display results

Close file

**Pseudocode to create course objects and store them in the appropriate data structure:**

Create insert function

If root is nullptr, set root to new node with course

Else

Invoke addNode function

Create addNode function

If course number is less than current node course number

If current node’s left child is nullptr

Add new node with course to node’s left child

Else

Recursively traverse node’s left sub tree

Else

If current node’s right child is nullptr

Add new node with course to node’s right child

Else

Recursively traverse node’s right sub tree

**Pseudocode to print out course information and prerequisites:**

Create printCourseInformation function

Get input for course ID

Assign current node to root

While current node is not equal to null

If course ID is equal to current node

Print course ID and course Name

If parameter is greater than two

For each parameter greater than two

Print prerequisite

If course ID is less than root

Set current node to left

Else

Set current node to right

**Menu Psuedocode**

Create integer choice for switch statement

Set choice to 0

While choice does not equal 4

Print “1. Load Data Structure”

Print “2. Print Course List”

Print “3. Print Course”

Print “4. Exit”

Switch (choice)

Case 1

Load courses

break

Case 2

Print course list

break

Case 3

Print course information

break

Case 4

End program

**Pseudocode that will print out the list of the courses in the Computer Science program in alphanumeric order**

**Vector**

Create integer i, j, and min

Create integer n and set equal to courses size

Create integer courseTemp

For I equals 0 and while j is less than n

Set min to i

For j equals i+1and while j is less than n

If min is greater than current course

Set min to j

Set courseTemp equal to course[i]

Set course[i] equal to min

Set min to courseTemp

For I equals 0 and while I is less than n

Print course number and course name

**Hashtable**

create vector sorting container

for each hash value

Use pushback to add each node of linked list to sorting container

Create integer i, j, and min

Create integer n and set equal to container size

Create integer courseTemp

For I is equal to 0 and while I is less than n

Set min equal to i

For j is equal to i+1 and while j is less than n

If min is less than current course number

Set min equal to current course number

Set courseTemp equal to current course number

Set current course number equal to min

Set min equal to courseTemp

Set head node for hash as null

For I is equal to 0 and while I is less than n

Add current course as node in linked list

For each hash value

For each node in linked list

Print course number and course name

**Tree**

Set node to root

Create function inOrder(node)

If node is equal to null

Return

Call function inOrder(node.left)

Print node course number and course name

Call function inOrder(node.right)

**Run-Time**

**Vector**



**Hash table**



**Tree**



**Explain the advantages and disadvantages of each structure in your evaluation.**

When it comes to data structures, there will be advantages and disadvantages for all. When creating a program you need to pick the data structure that works best for the application.

**Vector**

Some of the advantages of using a vector is that vectors are easy to implement which can in turn result in faster development of the code. Vectors can be an efficient option if you need to access elements sequentially, for example if you need to print the entire course list from top to bottom. Vectors also typically use less memory than other data structures, which could be important if memory is a top priority.

Some disadvantages of using a vector is that the insertion and deletion of elements within the vector could be slow due to the fact that it may require shifting elements. Another disadvantage would be searching in a vector is inefficient because it is going one by one through the vector and if the element you are looking for is either not present or the last element, it will have traveled through the entire vector.

**Hash table**

Some advantages of utilizing a hash table is that they on average provide a fast search which in turn will allow for quick search of our course information and number. Hash tables typically have a fast insertion and deletion which could be important if speed is you top priority.

Some disadvantages of utilizing a hash table is that there is no specific order of elements stored which could be important if you need to maintain the course order. Another disadvantage would be that handling collisions can be complex and cause performance degradation. Also, Hash table tend to us more memory when compared to vectors, so if memory is a top priority, hash table may not be the structure you should choose.

**Tree**

Advantages to using a tree (Binary Search Tree) is that they are based on an ordered structure that makes it easier for certain operations like printing a sorted list. Another advantage is that if a balanced tree is created, searching for specific elements will be efficient.

Some disadvantages of using a tree would be that they consume more memory than vectors do and if memory is top priority it can affect your choice. Trees are also more complex to implement than vectors or hash tables resulting in longer development.

**Recommendation**

For my recommendation I am recommending utilizing a vector structure. Utilizing a vector structure allows us to quickly sort the courses in alphanumeric order using quicksort and then printing the list out. Though this will have a slower response when finding and printing out a specific course and its prerequisites, the slower response will be negligible in the application we are designing. Using the vector will also allow us to develop the program faster allowing us to complete the project in a shorter amount of time. To help me come to this decision, I also utilized the Big O analysis completed above. All three structures ended up with O(N) for the runtime, but looking at the total cost columns, vector had less instances where something would run “n” times. There is more of a constant reliability with a vector.