# Project One

Camrin Stilwell

CS 300

6/10/2023

**Vector Pseudocode**

//Read data from file:

Open file with fstream

IF return value is -1, fine could not be found

ELSE file found

WHILE not the end of file

Read each line

IF less than two values return error message

ELSE read parameters

IF third or more parameters is elsewhere as first

Resume

ELSE return error message

Close file

//Create course objects:

Declare course vector

WHILE not the end of file

FOR each line in file

FOR first and second value

Use pushback to add value to vector

IF third value

Use pushback to add value up to new line

//Data Structure Search:

GET user input

IF input equals course number

PRINT course information

FOR each prerequisite

PRINT prerequisite course information

-------------------------------------------------------------------------------------------------------------

**Hash Table Pseudocode**

//Read data from file:

Open file with fstream

IF return value is -1, fine could not be found

ELSE file found

WHILE not the end of file

Read each line

IF less than two values return error message

ELSE read parameters

IF third or more parameters is elsewhere as first

Resume

ELSE return error message

Close file

//Create course objects:

Declare HashTable

Insert bid

WHILE not the end of file

FOR each line in file

FOR first and second value

Create temporary item for holding values

IF third value

Add to current value

//Print HashTable Search:

GET user input

Declare input as key

IF key is found

PRINT course information

FOR each prerequisite

PRINT prerequisite course information

ELSE

IF key not found

Display error message

Return

-------------------------------------------------------------------------------------------------------------

**Tree Pseudocode**

//Read data from file:

Open file with fstream

IF return value is -1, fine could not be found

ELSE file found

WHILE not the end of file

Read each line

IF less than two values return error message

ELSE read parameters

IF third or more parameters is elsewhere as first

Resume

ELSE return error message

Close file

//Create course objects:

Declare BinarySearchTree

Insert bids

WHILE not the end of file

FOR each line in file

FOR first and second value

Add coursed and courseName

IF third value

Add prerequisite until newline is found

//Print Tree Search:

GET user input

Declare search and print method

IF root is not null

Travel left

IF node matches coursed

PRINT course information

FOR each prerequisite

PRINT prerequisite course information

ELSE

Travel right

IF node matches coursed

PRINT course information

FOR each prerequisite

PRINT prerequisite course information

Return

-------------------------------------------------------------------------------------------------------------

**Menu Pseudocode**

//Display Menu:

WHILE user input does not equal 4

PRINT “1: Load Data”

PRINT ”2: Print Course List”

PRINT “3: Print Course”

PRINT “4: Exit”

GET user input

Switch case user input for menu

Case 1:

Load course data

Break;

Case 2:

Print course number and name

Break;

Case 3:

GET user input of course number

PRINT Course number, course name, and prerequisites

Break;

Case 4:

PRINT exit message

-------------------------------------------------------------------------------------------------------------

**Alphanumeric Order Pseudocode**

//Print courses in alphanumeric order:

SET mid as low + (high – low) / 2

SET pivot to courseName(mid)

WHILE courseName(low) is less than pivot

SET low as low + 1

ENDWHILE

WHILE pivot is less than courseName(high)

SET high as high – 1

ENDWHILE

IF low is greater than or equal to high

Return

ELSE create temporary variable as courseName(low)

SET courseName(low) as courseName(high)

SET courseName(high) to temporary variable

SET low as low + 1

SET high as high – 1

END IF

Return high

Main Function

Call sortList(courseName, 0, size – 1)

PRINT “List sorted”

FOR each course

PRINT course information

ENDFOR

End Main

-------------------------------------------------------------------------------------------------------------

**VECTOR**

| Code | Line Cost | # Times Executes | Total Cost |
| --- | --- | --- | --- |
| Open fstream | 1 | 1 | 1 |
| WHILE not at the end of the file, read each line | 1 | n | n |
| IF less than two values return error message | 1 | n | n |
| ELSE resume loop | 1 | n | n |
| Declare course vector | 1 | 1 | 1 |
| Loop file data | 1 | n | n |
| FOR each line in file | 1 | n | n |
| FOR first and second value, use pushback to add value to vector | 1 | n | n |
| IF third value, use pushback to add value up to new line | 1 | n | n |
| Total Cost | | | 8n + 2 |
| Runtime | | | O(n) |

**HASH TABLE**

| Code | Line Cost | # Times Executes | Total Cost |
| --- | --- | --- | --- |
| Open fstream | 1 | 1 | 1 |
| WHILE not end of the file | 1 | n | n |
| IF less than two values return error message | 1 | n | n |
| ELSE resume loop | 1 | n | n |
| Declare HashTable | 1 | 1 | 1 |
| Insert bid | 1 | n | n |
| Loop file data | 1 | n | n |
| FOR each line in file | 1 | n | n |
| FOR first and second value, create temporary item for holding values | 1 | n | n |
| IF third value, Add to current value | 1 | n | n |
| Total Cost | | | 9n + 2 |
| Runtime | | | O(n) |

**TREE**

| Code | Line Cost | # Times Executes | Total Cost |
| --- | --- | --- | --- |
| Open fstream | 1 | 1 | 1 |
| While not end of the file | 1 | n | n |
| If less than two values, return error message | 1 | n | n |
| ELSE resume loop | 1 | n | n |
| Declare Binary Search Tree | 1 | 1 | 1 |
| Insert bids | 1 | n | n |
| While not end of the file | 1 | n | n |
| Loop file data | 1 | n | n |
| FOR each line in file | 1 | n | n |
| FOR first and second value, add coursed and courseName | 1 | n | n |
| IF third value, add prerequisite until newline is found | 1 | n | n |
| Total Cost | | | 10n + 2 |
| Runtime | | | O(n) |

**Evaluation**

Vector:

An advantage of the vector structure is its fast file reading and inserting additional course objects. However, it is slow at searching for a particular course due to it needing to search through every course item.

Hash Table:

An advantage of the hash table is its fast searching as it uses keys to indicate items in the list. However, they are not sorted and need more time to sort through the course in alphanumeric order.

Tree:

An advantage of the Binary Search Tree is being able to sort items which makes it easier to find the searched item. However, this sorting is slower than the hash table but faster than the vector structure. A disadvantage of this would be the amount of additional time it would take to change anything.

**Recommendation**

According to the previously mentioned advantages and disadvantages, I would recommend using vectors for this project. Compared to the other two data structures, the vector structure had the lowest runtime of 8n +2. It is also fast at reading files and adding any objects that are needed. The speed that it takes to read and add objects makes up for the amount of time it takes to search for a course and print its corresponding information.