Mikaela Spence

CS 300

Project 1

**Vector Data Structure Pseudocode**

**Opening a File**

START

USE fstream to enable file opening

CALL the file

IF value = -1

DISPLAY “File not found”

ELSE

READ lines of file

IF values < 2

RETURN error

ELSE read values

CHECK that prerequisites exist as a course

CONTINUE until end of file

CLOSE File

**Creating Course Objects**

INITIALIZE course vector

READ file

LOOP through file contents

WHILE not End of File

FOR each line

PUSHBACK to add values to vector

CLOSE file

RETURN values

**Searching Data Structure**

OBTAIN user input for search

LOOP through vector data

IF input != courseNumber

CONTINUE search

IF input == courseNumber

PRINT course information

PRINT prerequisite information

END

**Hash Table Data Structure Pseudocode**

**Opening a File**

START

USE fstream to enable file opening

CALL the file

IF value = -1

DISPLAY “File not found”

ELSE

READ lines of file until EOF

IF values < 2

RETURN error

ELSE read values and parse into course ID, name and prereqs

CHECK that prerequisites exist as a course

CONTINUE until end of file

CLOSE File

**Creating Course Objects**

INITIALIZE course vector

CREATE Hash table

CREATE Insert method to be able to insert courses into Hash table

READ file

LOOP through file contents

WHILE not EOF

FOR each line

FOR 1st and 2nd values

CREATE temp variable to hold values

IF 3+ values exists

ADD to current value

CALL insert method for each value

CLOSE file

RETURN values

**Searching Data Structure**

OBTAIN user input for search

LOOP through vector data

IF input != courseNumber

CONTINUE search

IF input == courseNumber

PRINT course information

PRINT prerequisite information

END

**Binary Tree Data Structure Pseudocode**

**Reading File:**

USE fstream to be able to open file

OPEN file

IF the return value is “-1”, file is not found

ELSE file is found

WHILE not the EOF

READ each line

IF > 2 values

RETURN error

ELSE >= parameters

IF 3+ is in 1st parameter continue

ELSE return Error

CLOSE file

**Create Course Objects Structure**

INITIALIZE Course Structure

LOOP through file

IF not EOF

FOR each line

1st and 2nd values

ADD course Id and course name

If 3rd value exists

ADD Prereqs

**Creating the Tree and Adding Nodes:**

DEFINE Binary tree class

CREATE root = null

CREATE insert method

IF root = null

SET current course to root

ELSE

IF course # is < root

POINT left

IF left = null, ADD course

ELSE if course < move left

ELSE if course > move right

IF course # > root

POINT right

IF right = null, ADD course

ELSE if course < move left

ELSE if course > move right

**Search and Print from Binary Tree**

GET user input

CREATE print method

IF root != null

TRAVERSE left, output if found

TRAVERSE right, output if found

IF not found, return “course not found”

**Psuedocode for a Menu**

INITIALIZE variables (switch variable, course variables)

PRINT menu options

GET user input

WHILE user input != 4 (exit)

CASE 1:

LOAD course data from file

BREAK

CASE 2:

PRINT sorted course lists

BREAK

CASE 3:

OBTAIN user input for course number

PRINT course title, ID

BREAK

CASE 4:

EXIT

DEFAULT:

PRINT error message and repeat menu options

BREAK

**Printed List in Alphanumeric Order**

**Vector List**

CREATE print method for sorting

CREATE partition method

SET low indes for first element

SET high index to final element

SET midpoint to low index + (high index – low index)/2

SET pivot to midpoiny

WHILE low index < high index

SWAP low value to left, higher to right

SET temp value to low index

SET low index to high index

FOR each course in data structure

PRINT course information

**Hash Table List**

CREATE quicksort method

SET mid to 0, low to beginning, high to end

IF beginning >= end

RETURN

SET low index to partition

MAKE call to quicksort

CREATE display course method

LOOP through course list, incrementing up

PRINT each

**Binary Tree**

CREATE in order method

IF node != null

CHECK left side of tree first

PRINT from left-most

CHECK right side of tree

PRINT left to right

**Run Time Analysis**

**Vector Data Structure**

Creating vector O(N)

File lines O(1)

Adding course items O(1)

Printing O(N)

**Hash Table Data Structure**

Creating Hash table O(N)

File lines O(1)

Inserting O(1)

Printing O(1)

**Binary Tree Data Structure**

Creating Binary Tree O(nlogn)

Course lines O(1)

Inserting O(logn)

Printing O(logn)

**Advantages/Disadvantages**

**Vector**

Advantages: Simple to implement, less code, keeps insertion order, fast access to specific index. Very easy to add and remove the last element.

Disadvantages: Slow search, slow big O run time, not great for large data sets. Not easy to change elements in the middle. Not good for large data sets. Every item must be checked for a search, slowing search times significantly.

**Hash Table**

Advantages: Fast insertion and retrieval, works for larger data sets, fast access to specific index

Disadvantages: No set order, takes more memory, size is better to be known in advance. May have too many collisions which can make less efficient.

**Binary Search Tree**

Advantages: Elements are already sorted, very efficient sort and search times

Disadvantages: More complex, slower insertion and deletion overall. If trees are not balanced, may take more time.

**Recommendations**

The listings of courses and prerequisites for a college are likely to be large in number. Course IDs are likely to be more like 12345 than 10, and not start at 00000. This makes the vector data structure a little less capable than the other two. There are too many courses available and vector sorting is not as efficient as hash or binary trees. With a hash table, it is helpful to know the amount of data in advance, which we can for this project, although the number of courses may increase over time, or change and hash tables are less adaptable to this. Looking at the runtimes, the binary search tree is over all the fastest across several factors. It works well with a large data set, may take a little longer to insert than hash tables, but makes up for it with quicker searches and printing. Unfortunately, this is the structure in the course I had the most confusion with.