Jerry Barboza

Week 6

CS 300

Project One

1)

Vector Data Structures

Begin

```
FUNCTION ReadFile(filename)
        OPEN file
        For file not empty
                read_line(file)
                parse each line
                check for errors
                IF no errors
                        Exit file
        End Function

FUNCTION CourseObj (objects)
        Initialize variables for courses
        READ File
        WHILE file is open
                Store the course obj in a vector data structure
        End Function

FUNCTION SearchData(data)
        Initialize variables for opening file
        Open file
        WHILE file is open
                OUTPUT: course information
                Store data gathered in a data structure
        End of Function
```

END

4-3 Milestone: Hash Table Data Structure Pseudocode

Begin

```
        FUNCTION ReadFile( file, lines)
                WHILE open file
                        Read the data THEN
                        Parses each line
                IF no errors in line THEN
                        Move to next line
                IF error THEN
                OUTPUT: prints the error
                Return
        END Function

        FUNCTION CourseObject(course, file)
                FOR every line
                IF course in line THEN
                        Append new course to courses
                ELSE
                        Move to next line
        End Function

    FUNCTION PrintInfoPrer(courses , info)
        Open File
        courses =< vector>
        IF courses  is empty
                OUTPUT: “No courses in file”
        ELSE
                OUTPUT: (courses, info)
        Return
    End Function
```

END

Tree Data Structure Pseudocode

Begin

```
FUNCTION ReadFile( file, lines)
        WHILE open file
                Read the data THEN
                Parses each line
        IF no errors in line THEN
                Move to next line
        IF error THEN
        OUTPUT: prints the error
        Return
End FUNCTION


FUNCTION CourseObject(course, file)
        FOR every line
        IF course in line THEN
                Append new course to courses
        ELSE
                Move to next line
        End FUNCTION


FUNCTION PrintInfoPrer(courses , info)
    Open File
    courses =< vector>
    IF courses  is empty
            OUTPUT: “No courses in file”
    ELSE
            OUTPUT: (courses, info)
    Return
End Function
```

End

2)

Pseudocode for Menu:

```
Begin
    Menu()
        OUTPUT: “Load Data Structure”
        OUTPUT: “Course List”
        OUTPUT: “Course”
        OUTPUT: Exit

        SWITCH():

        Case 1:
            loadDataStructure
            break
        Case 2:
            Print Course List in alphanumerically order
            Break
        Case 3:
            Print Course;
            Break
        Case 4:
            Exit
            Break
```

END

3) Pseudocode that will print out the list of the courses in the Computer Science in alphanumeric order.

Begin

```
FUNCTION SortCourseAlphanumeric(courses)
    WHILE File is open
        ListCourses = string(i) for i in ListCourse
        Sort ListCourse from low to high
        PRINT: sorted ListCourses
```

END

EVALUATION:

4) **FUNCTION ReadFile(file, lines)**

Code	Line Cost	# Times Executed	**Total Cost**
WHILE open file	1	n	n
Read the data THEN	1	1	1
Parses each line	1	1	1
IF no errors in line	1	n	n
Move to next line	1	1	1
IF error THEN	1	n	n
Print the error	1	1	1
		TOTAL COST	T(n) = 3n + 4
		Runtime	O(n)

FUNCTION CourseObject(course, file)

Code	Line Cost	# Times Executed	**Total Cost**
FOR every line	1	n	n
IF course in line THEN	1	n	n
Append new course to courses	1	1	1
ELSE move to next tine	1	n	n
		TOTAL COST	T(n) = 3n + 1
		Runtime	O(n)

FUNCTION PrintInfoPrer(courses , info)

Code	Line Cost	# Times Executed	**Total Cost**
IF courses is empty	1	n	n
RRINT: “courses is empty”	1	1	1
ELSE	1	n	n
Print: course information	1	1	1
Return	1	1	1
		TOTAL COST	T(n) = 2n + 3
		Runtime	O(n)

Now adding all three functions of the pseudocode we get:

T(n) =(2n + 3) + (3n +1) + (3n +4) = 8n + 8 therefore a *runtime* of ***O(n)***.

5) Hash tables is a data structure that stores unordered items mapping each item to a location in an array. The main advantage of hash table is that the time complexity for searching or inserting/removing an item is O(1). This is faster than searching a list at O(N) or a binary search with a time complexity of O(log N). A vector is great when the order of the list doesn't matter also since a vector has time complexity of O(1). When adding to the list you can just append and when removing the that item you just added you can just "pop_back()".

The benefits of BST is that an N-node binary tree's height may be as small as O(logN), making extremely fast searches. The advantages of BST is that we can obtain all keys in sorted order by just doing Inorder Traversal of BST. Also order statistics, finding lower and greater elements and doing range queries are easier to do with BST than Hash. BST is also easy to implement compared to hashing. Even though the operations for Hash average time is O(1), some operations can be very costly like when resizing the table all. Therefore BST can be better since all operations are guaranteed to work in O(log N).

6) Out of the three, I would consider Trees the best however for this project the hash table can have an advantage over Trees since we are mostly inserting, deleting and searching and for these, Hash has an advantage since the time complexity is O(1). Once we get towards the end, we have to sort the computer science courses and when it comes to

sorting, Trees would be the best option since BST can get the keys sorted with in-order traversal. Hashing can cost O(n^2) when resizing occurs therefore making it slower to run. Overall BST are memory efficient and Hash table is not.

Citations:

Zybooks

https://www.geeksforgeeks.org/advantages-of-bst-over-hash-table/