Brandon R Mitchell

Project 1

CS 300

Pseudo Code for Milestone 1 Vector: I have nothing to correct here as I was extremely late with my submission so I will try my best as to what I think is correct. As my corrections for my mod 5 assignment, I was informed that I should have white space and indentation for blocks of code.

ABC University

Load text parsing libraries and headers

Define a struct to hold course data

struct Course {}

courseID

courseName

preCount

preList

Course() (constructor) {courseID = courseName =””; preCount = 0; preList = “”}

Main()

Create new List named courseList of the struct-type Course

Get CSV file path from user

If no data passed use default location

Call txtParser() passing CSV file path

Call validateList() passing courseList

Get user value to search for and Store in userSearch

Call printCourse() passing userSearch

End

txtParser(String)

Create a local List named tempList

Open file found at the path in String by invoking parser libraries

Loop row by row until end of file (eof)

If first and second string are present

Add the first String to struct at courseID

Add the second String to Struct at courseName

Loop until file handler has no value in a column (indicates no more prerequisite)

Increment a variable named preCount for each prerequisite found

Concatenate a localString named preNames for each prerequisite

Add preCount to struct at preCount

Add preNames to struct at preList

Return tempList

End

searchList(String)

Create tempCourse of type Course

Loop through list For Each Course

If String is the same as courseID

Set tempCourse to Course

Return tempCourse

End

printCourse(String)

Create tempCourse of type Course

Set tempCourse equal to searchList(String)

Output courseID to console

Output courseName to console

Loop 0 to preCount

For each Course in preList

Call printCourse() passing preList

End

**Milestone 2 Hash Table**

My corrections here are I have now corrected the course object pseudocode as previously it was not present, cleared up my lines of communications which negatively impacted understanding

Load text parsing libraries and headers

Define a struct to hold course data

struct Course {}

courseID

courseName

preCount

prelist

Course() (constructor) {courseID = courseName = ””; preCount = 0; preList = “”}

Class HashTable{}

-struct bucket

Course

key

next pointer

+hash()

+printAll()

+List<> hashTable

Main()

Create new List named courseList of the struct-type CourseMap

Get CSV file path from user

If no data passed use default location

Call txtParser() passing CSV file path

Call validateList() passing courseList

Get user value to search for and Store in userSearch

Call printCourse() passing userSearch

End

txtParser (String)

Open file found at the path in String by invoking parser libraries

Loop row by row until end of file (eof)

If first and second string are present

Call hash passing the first string

Add to struct at hash position within tempList

Add the first String to struct at courseID

Add the second String to Struct at courseName

Loop until file handler has no value in a column (indicates no more prerequisite)

Increment a variable named preCount for each prerequisite found

Concatenate a localString named preNames for each prerequisite

Add preCount to struct at preCount

Add preNames to struct at preList

Return tempList

End

searchList(String)

Create tempCourse of type bucket

Set tempCourse to the bucket at the hash location of String

Loop through list For Each Course

If String is the same as courseID

Set tempCourse to Course

Return tempCourse

End

printCourse(String)

Create tempCourse of type bucket

Set tempCourse equal to hash (String)

Loop through all chained buckets at tempCourse

Output courseID in Course struct found within tempCourse to console

Output courseName in Course struct found within tempCourse to console

Loop 0 to preCount

For each Course in preList

Call printCourse() passing preList

End

validateList()

Create tempCourse of type bucket

Create variable valid and Set to True

For Each Course

If valid is False break

While tempCourse next is not null

Loop 0 to preCount

Set tempCourse equal to searchList(preList token)

If tempCourse courseID is empty Set valid to False

Return valid

End

int Hash(key)

# to be decided; how do we want to hash the String courseID

# Hash function needs to be cognizant that the first 4 letters

validateList()

Create tempCourse of type Course

Create variable valid and Set to True

For Each Course

If valid is False break

Loop 0 to preCount

Set tempCourse equal to searchList(preList token)

If tempCourse courseID is empty Set valid to False

Return valid

End

**Milestone 3 Binary search tree**

Here I had minimal correction

struct Course {}

courseID

courseName

preCount

prelist

Course() (constructor) {courseID = courseName = ””; preCount = 0; preList = “”}

Class BinaryTree{}

-struct Node

Course

right pointer

left pointer

-root

+printCourse()

+BinaryTree()

Main()

Create new BinaryTree named courseTree of the struct-type Course

Get CSV file path from user

If no data passed use default location

Call txtParser() passing CSV file path

Call validateList() passing courseTree

Get user value to search for and Store in userSearch

Call printCourse() passing userSearch

End

txtParser (String)

Open file found at the path in String by invoking parser libraries

Loop row by row until end of file (eof)

If first and second string are present

Add the first String to struct at courseID

Add the second String to Struct at courseName

Loop until file handler has no value in a column (indicates no more prerequisite)

Increment a variable named preCount for each prerequisite found

Concatenate a localString named preNames for each prerequisite

Add preCount to struct at preCount

Add preNames to struct at preList

Return tempList

End

searchList(String)

Create tempCourse of type Node

Set tempCourse to the bucket at the hash location of String

Loop through list For Each Course

If String is the same as courseID

Set tempCourse to Course

Return tempCourse

End

printCourse(String)

Create tempCourse of type bucket

Set tempCourse equal to root

Loop until tempCourse is Null

If the Node at tempCourse contains a bidId equal than to String

Output courseID in Course struct found within tempCourse to console

Output courseName in Course struct found within tempCourse to console

Loop 0 to preCount

For each Course in preList

Call printCourse() passing preList

If the Node at tempCourse contains a courseID less than to String

Set tempCourse equal to the left Node

If the Node at tempCourse contains a courseID greater than to String

Set tempCourse equal to the right Node

End

validateList()

Create tempCourse of type Node

Create variable valid and Set to True

For Each Course

If valid is False break

While tempCourse next is not Null

Loop 0 to preCount

Set tempCourse equal to searchList(preList token)

If tempCourse courseID is empty Set valid to False

Return valid

End

(4) RUN TIME

Loading data > Vector = O(1), Hash table = O(1) – O(N), Binary tree = O(logN)

Search > Vector = O(n), Hash table = O(1) – O(N), Binary tree = O(log N) – O(n)

Print > Vector O(n log n), Hash table = O(n), Binary tree = O(n)

(5) Advantages and disadvantages

Hash table will always run in O(1) which means that is runs at its largest, meaning it’s the safest and most secure. Disadvantage is memory is not forever which means this table will eventually have to endure some issues where it may have to run before O(n).

Binary tree will constantly run at logN which will determine how the data is interpreted and then re emitted. Disadvantages is we will have to sort the data, which ultimately bogs the binary tree down to where it will only run at O(n).

(6) Recommendations

Since we are making recommendations based on speed, performance, security and functionality, for this assignment I will be recommending hash table due to its speed, functionality in O(1), hash table is superior.