Dayne R Firth II

4/12/2025

Project 1

Vector Pseudocode

Void openFile(){

OPEN file “course\_data.txt”

IF file cannot be open

PRINT “ERROR: file cannot be opened.”

RETURN to menu and DISPLAY Error

ELSE

CALL readFile()

}

Void readFile(){

READ file “course\_data.txt”

IF file cannot be read

PRINT “ERROR: file cannot be read.”

RETURN to menu DISPLAY Error

ELSE

CALL parseCourse()

}

Void parseCourse(){

LOOP through entire document until end of document

WHILE not at the end of the document

READ line

SPLIT line by comma into LIST lineData

IF LENGTH of LineDATA < 2

PRINT “ ‘<lineData>’ is formatted incorrectly”

MOVE to next line

ELSE

SET courseNumber = lineData[0]

SET courseName = lineData[1]

IF lineData[2] = ‘null’

SET prerequisite1 = ‘NONE’

MOVE to next line

ELSE SET prerequisite1 = lineData[2]

IF lineData[3] = ‘null’

MOVE to next line

ELSE SET prerequisite2 = lineData[3]

ENDWHILE

CLOSE file

}

Void searchCourse(Vector<Course> courses, String courseNumber) {

for all courses

if the course is the same as courseNumber

print out the course information

for each prerequisite of the course

print the prerequisite course information

}

Hash Table Pseudocode

Void hashTable (){

CREATE new hash table

INITIAL size enough for data set to be imported

}

Void readFileHash(){

READ file “course\_data.txt”

IF file cannot be read

PRINT “ERROR: file cannot be read.”

RETURN to menu DISPLAY Error

ELSE

CALL parseCourseHash()

}

Void parseCourseHash(){

LOOP through entire document until end of document

WHILE not at the end of the document

READ line

SPLIT line by comma into LIST lineData

IF LENGTH of LineDATA < 2

PRINT “ ‘<lineData>’ is formatted incorrectly”

MOVE to next line

ELSE

SET courseNumber = lineData[0]

SET courseName = lineData[1]

IF lineData[2] = ‘null’

SET prerequisite1 = ‘NONE’

MOVE to next line

ELSE SET prerequisite1 = lineData[2]

IF lineData[3] = ‘null’

MOVE to next line

ELSE SET prerequisite2 = lineData[3]

CALL insertCourseHashTable()

ENDIF

ENDWHILE

CLOSE file

}

Void insertCourseHashTable(){

ADD courseNumber, courseName, prerequisite1, prerequisite2 to hashtable

IF space is not available in hash table THEN

RESIZE hash table

ENDIF

}

void searchCourse(HashTable<Course> courses, String courseNumber) {

}

Binary Search Tree Pseudocode

Void binarySearchTree (){

CREATE new Binary Search Tree

INITIAL root is null

}

Void readFileBST(){

READ file “course\_data.txt”

IF file cannot be read

PRINT “ERROR: file cannot be read.”

RETURN to menu DISPLAY Error

ELSE

CALL parseCourseBST()

}

Void parseCourseBST(){

LOOP through entire document until end of document

WHILE not at the end of the document

READ line

SPLIT line by comma into LIST lineData

IF LENGTH of LineDATA < 2

PRINT “ ‘<lineData>’ is formatted incorrectly”

MOVE to next line

ELSE

SET courseNumber = lineData[0]

SET courseName = lineData[1]

IF lineData[2] = ‘null’

SET prerequisite1 = ‘NONE’

MOVE to next line

ELSE SET prerequisite1 = lineData[2]

IF lineData[3] = ‘null’

MOVE to next line

ELSE SET prerequisite2 = lineData[3]

CALL insertCourseBST(courseNumber, courseName, prerequisite1, prerequisite2)

ENDIF

ENDWHILE

CLOSE file

}

Void insertCourseBST (courseNumber, courseName, prerequisite1, prerequisite2){

CREATE new nod with course data

IF BST root is null

SET root to new node

ELSE

CALL addnode(root, new node)

ENDIF

}

Void addNode (currentNode, newNode){

IF new course number is less than current node course

IF current left node is null

Set current node to new node

ELSE

CALL addNode with current node left and new node

ELSE

IF current right node is null

Set current node to new node

ELSE

CALL addNode with current node right and new node

}

void searchCourse(Tree<Course> courses, String courseNumber) {

}

Menu Pseudocode

BEGIN

DISPLAY main menu

WHILE users does not exit program

GET user choice

SWITCH depending on choice

CASE 1: Load course data

START timer

READ the CSV file

FOR each bid in the CSV file

IF using vector for database

STORE bids in a vector

ELSE IF using vector for hash table

STORE bids in a hash table

ELSE using vector for binary search tree

STORE bids in the Binary Search Tree

END for

END timer

DISPLAYS the number of bids loaded and the time it took

CASE 2: Display all course alphanumerically for the CS Department

IF using vector

CALL displaySortedVector(courseList)

ELSEIF using a hash table

CALL displaySortedHastTable(courseTable)

ELSE using a binary search tree

CALL displaySortedBinarySearchTree(courseBST)

CASE 3: Find Course and Prerequisites

INPUT course number

SEARCH dataset for course number

IF course number is found

DISPLAY Course Details

IF prerequisites none

RETURN

ELSE

DISPLAY prerequisites

ELSE

DISPLAY “No Course data Found”

CASE 9: Exit program

PRINT Goodbye

END SWITCH

END WHILE

END

Computer Science program in alphanumeric order Pseudocode

//Vector Data set

Void displaySortedVector(courseList)

SORT courseList by courseNumber in alphanumeric order

FOR each course in courseList

DISPLAY courseNumber and courseName

END FOR

Void displaySortedHastTable(courseTable)

CREATE empty list for store sorted course (courseStore)

LOOP thru courseTable

WHILE not at the end of data set

FOR each course in the courseTable

ADD to course to courseStore

ENDFOR

ENDWHILE

SORT courseList by courseNumber in alphanumeric order

FOR each course in courseList

DISPLAY courseNumber and courseName

END FOR

Void displaySortedBinarySearchTree(courseBST)

IF courseBST is null

RETURN

TRAVERSE the tree in order

FOR EACH node

DISPLAY courseNumber and courseName

END FOR

Evaluation

Vector Data Base

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| Read Line from the file | 1 | n | 1 |
| Split line by comma | 1 | n | 1 |
| Check if the line is Formatted Correctly | 1 | n | 1 |
| Set courseNumber = lineData[0] | 1 | n | 1 |
| Set courseName = lineData[1] | 1 | n | 1 |
| Check if lineData[3] is not Null | 1 | n | 1 |
| IF lineData[3] is not Null Set prerequisite 1 | 1 | n | 1 |
| Check if lineData[4] is not Null | 1 | n | 1 |
| IF lineData[4] is not Null Set prerequisite 2 | 1 | n | 1 |
| Add course Details to vector data set | 1 | n | 1 |
| **Total Cost** | | | 10n+n |
| **Runtime** | | | O(n) |

Hash Table

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| Read Line from the file | 1 | n | 1 |
| Split line by comma | 1 | n | 1 |
| Check if the line is Formatted Correctly | 1 | n | 1 |
| Set courseNumber = lineData[0] | 1 | n | 1 |
| Set courseName = lineData[1] | 1 | n | 1 |
| Check if lineData[3] is not Null | 1 | n | 1 |
| IF lineData[3] is not Null Set prerequisite 1 | 1 | n | 1 |
| Check if lineData[4] is not Null | 1 | n | 1 |
| IF lineData[4] is not Null Set prerequisite 2 | 1 | n | 1 |
| Call insertCourseHashTable | 1 | n | n |
| Add course Details to Hash Table | 1 | n | 1 |
| IF space isn’t not available resize Hash Table | 1 | n | 1 |
| **Total Cost** | | | 11n+2n |
| **Runtime** | | | O(1) |

BST Table

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| Read Line from the file | 1 | n | 1 |
| Split line by comma | 1 | n | 1 |
| Check if the line is Formatted Correctly | 1 | n | 1 |
| Set courseNumber = lineData[0] | 1 | n | 1 |
| Set courseName = lineData[1] | 1 | n | 1 |
| Check if lineData[3] is not Null | 1 | n | 1 |
| IF lineData[3] is not Null Set prerequisite 1 | 1 | n | 1 |
| Check if lineData[4] is not Null | 1 | n | 1 |
| IF lineData[4] is not Null Set prerequisite 2 | 1 | n | 1 |
| Call insertCourseBST | 1 | n | n |
| CREATE new node | 1 | n | 1 |
| IF BST root is null set root as new node | 1 | n | 1 |
| ELSE CALL addnode | 1 | n | n |
| IF new courseNumber is less than current node | 1 | n | 1 |
| IF new left node is null that set node to course | 1 | n | 1 |
| ELSE Call addnode with current node left | 1 | n | n |
| ELSE if right node is null set current node to course | 1 | n | 1 |
| ELSE Call addnode with current node right | 1 | n | n |
| **Total Cost** | | | 14n+4n |
| **Runtime** | | | O(log n) |

Each data structure has its own advantages and disadvantages and one should be selected based on which best meets the needs of the application. Vector data structure is easy to create and maintain. It is easy to add data to the dataset by adding an additional line at the end O(1). On the other hand, it is slow to search for a specific item like a course number, runtime is O(n). The hash table is slightly more complex to implement than a vector, but offers highly efficient search performance, with a typical runtime of O(1) for lookups. The downside of using a hash table is that if it is not maintained properly it can lead to higher memory overhead and degraded performance. Additionally, because hash tables are unordered, displaying data in a sorted manner requires extracting all items, sorting them, and then displaying those results in a runtime of O(n log n) for sorted output. A binary search tree (BST) is sorted by default, so needing to display a sorted list has a runtime of O(n). Searching and inserting into a balanced BST is also efficient, with a runtime of O(log n). However, if the tree becomes unbalanced, performance degrades to O(n). Implementing a well-balanced BST is more complex compared to a vector or a hash table.

My recommendation is that the school uses a hash table data structure because it provides the best lookup performance compared to the other data structures with an average runtime of O(1). Sorting and displaying a data set in alphanumeric order in a hash table requires a runtime of O(n log n) which is manageable. The memory usage of a hash table is much lower than a binary search tree and the hash table is easier to maintain. Especially since loading course information will be generally a one-time event with occasional updates. Vector data set is easier, but the ability to search significantly faster with a hash table outweighs it’s simplicity. Again, my recommendation is a hash table because it provides the best balance of speed, efficiency, and practicality for the school’s requirements.