Stephen Cessna

11/30/2022

CS-300 Project One

**Vector Pseudocode**

INT numPrerequisitesCources(Vector<Course>courses string courseNum){

totalPrerequisites = prerequisites size

FOR each prerequisite in totalPrerequisites

PRINT/ADD number of totalPrerequisites

}

void printCourseInformation(Vector<Course> courses, String courseNumber){

IF course is equal to courseNumber

OUTPUT courseInformation

OUTPUT prerequisites

}

OPEN file

parse through file

IF line size = 2

Set courseName = to line [0]

set courseNumber = to line[1]

return newCourse

IF line size = 3

set courseName = to line [0]

set courseNumber = to line[1]

set coursePrerequsite = to line[3]

return newCourse

IF line size = 4

set courseName = to line [0]

set courseNumber = to line[1]

set coursePrerequsite = to line[3,4]

return newCourse

IF line size < 2

OUTPUT error in format of file

| **Vector** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
|  | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 1 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 1 | n | n |
| **Total Cost** | | | 4n + 1 |
| **Runtime** | | | O(n) |

**Hash Table Pseudocode**

string courseName

string courseNumber

vector<string>prerequisite

INT numPrerequisitesCourses(Hashtable<Course>courses string courseNum){

create a key for the courseNumber

retrieve node using key set to new node

WHILE node != nullptr

IF node -> course is equal to courseNumber

totalPrerequisites = node prerequisites size

FOR each prerequisite in totalPrerequisites

PRINT number of totalPrerequisites

ELSE node = node->next

}

void printCourseInformation(Hashtable<Course> courses, String courseNumber){

create a key for the courseNumber

retrieve node using key set to new node

WHILE node != nullptr

IF node -> course is equal to courseNumber

OUTPUT courseInformation

OUTPUT prerequisites

}

OPEN file

parse through file

IF line size = 2

Set courseName = to line [0]

set courseNumber = to line[1]

return newCourse

IF line size = 3

set courseName = to line [0]

set courseNumber = to line[1]

set coursePrerequsite = to line[3]

return newCourse

IF line size = 4

set courseName = to line [0]

set courseNumber = to line[1]

set coursePrerequsite = to line[3,4]

return newCourse

IF line size < 2

OUTPUT error in format of file

| **Hash Table** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Create key for courseNumber** | 1 | n | n |
| **Retrieve node using key set to new node** | 1 | n | n |
| **While node != nullptr** | 1 | n | n |
| **If node->course = courseNumber** | 1 | n | n |
| **totalPrerequisites = node prerequisites size** | 1 | n | n |
| **For each prerequisite in totalPrerequisites** | 1 | n | n |
| **Print num of totalPrerequisites** | 1 | 1 | 1 |
| **Else node -> next node** | 1 | n | n |
| **While node != nullptr** | 1 | n | n |
| **If node -> course = courseNumber** | 1 | n | n |
| **Print courseInformation** | 1 | 1 | 1 |
| **Print coursePrerequisites** | 1 | 1 | 1 |
| **Total Cost** | | | 9n + 3 |
| **Runtime** | | | O(n) |

**Binary Search Tree Pseudocode**

STRING courseName

STRING courseNumber

vector<string>prerequisite

INT numPrerequisiteCourses(Tree<Course> courses) {

WHILE current != ptr

IF current->course is equal to courseNumber

totalPrerequisites = current prerequisites size

FOR each prerequisite in totalPrerequisites

PRINT total number of totalPrerequisites

ELSE current = current->right

}

VOID printCourseInformation(Tree<Course> courses, String courseNumber){

WHILE current != null ptr

IF current course is equal to courseNumber

OUTPUT courseInformation

OUTPUT coursePrerequisites

}

OPEN file

PARSE through file

IF line size = 2

Set courseName = to line [0]

set courseNumber = to line[1]

return newCourse

IF line size = 3

set courseName = to line [0]

set courseNumber = to line[1]

set coursePrerequsite = to line[3]

return newCourse

IF line size = 4

set courseName = to line [0]

set courseNumber = to line[1]

set coursePrerequsite = to line[3,4]

return newCourse

IF line size < 2

OUTPUT error in format of file

| **Binary Search Tree** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **While current != nullptr** | 1 | n | n |
| **If current course = courseNumber** | 1 | n | n |
| **totalPrerequisites = current prerequisites** | 1 | n | n |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 1 | 1 | 1 |
| **Else node becomes right** | 1 | n | n |
| **While current != nullptr** | 1 | n | n |
| **If currentCourse = courseNumber** | 1 | n | n |
| **Print courseInformation** | 1 | 1 | 1 |
| **Print coursePrerequisites** | 1 | 1 | 1 |
| **Total Cost** | | | 7n + 3 |
| **Runtime** | | | O(n) |

**Menu Pseudocode**

INITIALIZE choice = 0

INITIALIZE choiceTwo = 0

WHILE(choice != 9)

OUTPUT “Menu:”

OUTPUT “ 1. Load Data Structure”

OUTPUT “ 2. Print Course List”

OUTPUT “ 3. Print Course”

OUTPUT “ 9. Exit”

OUTPUT “ What would you like to do?”

SWITCH(choice)

case 1:

loadCourses(fileName, courses)

break;

case 2:

IF(courses != nullptr)

OUTPUT courseList

break;

case 3:

IF(courses != nullptr)

OUTPUT newCourse

break;

OUTPUT “Thank you for using the course planner!”

**Advantages and Disadvantages**

For a vector some advantages would be that you have the ability to add or remove items in the vector throughout the program. You also have the ability to change the size of the vector and multiple data types can be stored. Now some disadvantages would be the time that it would take to run through the vector. Another disadvantage of a vector would be the amount of memory consumption compared to the other lists. For a hash table some advantages would be that hash tables are more efficient than other search trees however the hash table could take up more space than is needed and hash tables do not preserve the other of the list. For binary search tree advantages one would be that a binary search tree does retrieve items in order. Also, a binary search tree would be able to access the list faster. A disadvantage for the binary search tree would be that you have to maintain balance for the best performance.

**Recommendation**

For this assignment I would recommend using a binary search. I would choose the binary search tree over the vector because the binary search tree would be able to out preform the vector. The reason that I would choose the binary search tree over the hash table is because the hash table wouldn’t provide an ordered list of data and would need resorted. Since this assignment requires displaying the list and the data in alphanumeric order, I would recommend using the binary search tree.