Danielle Sousa

CS-300 Analysis and Design

April 6th,2023

**Project One**

**Vector Pseudocode**

// Open file:

Vector<string> openReadFile (string filename) {

Initialize vector <string>;

Initialize to hold a single line;

Initialize fstream to get file contents;

Open the file with fstream by using the file name;

IF (return value > -1) {

OUTPUT “File not found”;

}

ELSE file is found

If not end of file (EOF){

READ each line;

PARSE each line;

IF less than 2 values in a line

OUTPUT “Error”;

Else READ parameters;

CLOSE the file;

}

//Create course for each line and store in vector:

Vector <Course> createSchedule (vector <string> contents){

Initialize vector <Course>;

Initialize string fstream linestream;

Initialize string token to store each word from the line;

Initialize int count to track tokens per line;

}

For (new Course){

SET count = 0;

create new Course course ;

Fill linestream with contents;

Pull token from linestream until end of line;

}

IF (count == 0){

course courseNum = token

++ count

}

ELSE IF (count ==1){

courseName = token

++ count

}

ELSE IF (token = courseNum in Use){

Push back token to PreReqs;

ELSE

OUTPUT “ Prerequisite required for this course”

++ count;

If(count < 2){

OUTPUT “ Error with file. Check course number and name.”;

Empty lineStream;

Push course to back;

}

RETURN courses;

Class Course {

Int courseNum;

String courseName;

Vector<string> prerequisites

}

// Course Search:

CourseSearch (vector <Course> courses, string courseNum){

Create empty Course;

FOR( each course){

IF(currentCourseNum == courseNum)

RETURN course;

}

RETURN empty;

Void Print(vector< Course> courses, string courseNum){

Create Course Object;

IF( course return empty){

OUTPUT “ Course not in schedule”;

ELSE IF

OUTPUT “ course courseNum and courseName”;

}

FOR( prerequisites )

OUTPUT “ prerequisites”

}

**Hash Table Pseudocode**

//Read File

Use fstream to open file

Method to loadCourses(string csvPath,dataStructure)

IF value is -1

File not found

Else

File found

While != EOF

Read each line

IF less than 2 values in a line

OUTPUT “Error”;

Else READ parameters;

IF 3 or more parameters

IF 3 or more parameter is in first parameter

Continue

ELSE return Error

CLOSE the file;

}

// HashTable

INITIALIZE Course Vector vector <Node> nodes

Create HashTable Class

Create method to insert items into HashTable

Loop file

While != EOF

FOR each line

FOR first amd second values

Create temp items to hold values

IF 3rd value

ADD to currValue

Call method for each value

//Search HashTable

Get input for Course ID

Assign key to course ID

//Print HastTable

IF key found

PRINT Course information

PRINT Prerequisite Course Information

ELSE

While node !- null check against key

IF key matches CourseID

RETURN Course

PRINT Course Information

Next Node

**Tree Pseudocode**

//Read File

Use fstream to open file

Method to loadCourses(string csvPath,dataStructure)

IF value is -1

File not found

Else

File found

While != EOF

Read each line

IF less than 2 values in a line

OUTPUT “Error”;

Else READ parameters;

IF 3 or more parameters

IF 3 or more parameter is in first parameter

Continue

ELSE return Error

CLOSE the file;

}

INITIALIZE Course Vector vector <Node> nodes

DEFINE BinaryTree Class

Create root to point to null

Create Method for Insert

IF root is null, Current Course is Root

ELSE

IF left is null,

ADD Course number

ELSE

IF Course number is less than

ADD to left

ELSE

IF Course number is greater

ADD right

ELSE

IF Course number is greater than Root

ADD right

IF right equal null

ADD course ID

ELSE

IF Course number is less than

ADD left

ELSE

IF Course number is greater

ADD right

Search Tree

GET input

Create Method for Print

IF root != null

Traverse Left give output IF found

ELSE

Traverse Right give output IF found

**Menu Pseudocode**

While selection != 9

{

Print “1. Load Data”

Print “2. Print Course List”

Print “3. Print Course”

Print “ 9. Exit”

Switch User Menu Input

{

User Inputs “1”

Loads Course Data

Prints ”Make Another Selection”

User Input “2”

Prints Course Name and number

Prints “ Make Another Selection”

User Input “3”

Prints “ Enter Course Number for search”

User Input Course Number

Prints Course Number, Course Name, Prerequisites Course Numbers

User Input “9”

Prints “Exit Program. GoodBye”

**}**

**Alphanumeric Order Of courses Pseudocode**

Create method for Sort print

Print Courses Sorted

Create method Partition

Int Partition

Set low to first element

Set Mid to low index

Set high to last element

Create method QuickSort

quicksort

Set low to Begin

Set high to end

If begin >= end

Return

Recursive call to quicksort

Create Method Display Course

Void displayCourse{

Loop vector to display courses

Display Courses

Create method inOrder

Void binarySearchTree in Order{

If node != null

Check left leaf

Node -> left

Check right leaf

Node => Right

}

}

**Pro and Cons of each Data Structure**

**Vector Data Structures :**

**Advantages:**

1. Scalability- they can handle large amounts of complex data

2. Speed- can search for similarities in data much faster than traditional databases, allowing real time analysis

3. Flexibility- can store and manage a large range of data types.

**Disadvantages:**

1. Not ideal for inserting or deleting elements in the middle since the rest of the elements would require adjustments.
2. Not ideal due to large amounts of data stored in memory and resulting in lower loading times**.**

**Hash Table**

**Advantages:**

1. Better Synchronization
2. More efficient
3. Provides constant time for searching.
4. Can automatically resize itself.
5. Easy to use.
6. Efficient insertion and deletion.
7. Can store any Data Type.

**Disadvantages:**

1. Inefficient when there are many collisions.
2. Does not allow Null values.
3. Can be complex to implement.
4. Does not maintain the order of elements.
5. Limited amount of capacity

**Binary Tree**

**Advantage:**

1. Efficient searching- each node has at most two child nodes, that allow for binary search algorithms to be used. So search operations can be performed in 0(log n) time.
2. Ordered Traversal- This allows operations to be performed on the nodes in a specific order.
3. Memory Efficient- because they only require two child pointers per node.
4. Fast insertion and deletion- Can be used to perform insertion and deletions at 0(log n) time complexity. The obvious choice for applications requiring dynamic data structures.
5. Easy to implement.
6. Useful for sorting - used to sort and implement sorting algorithms such as heap sort and binary tree sort.

**Disadvantages:**

1. Limited Structure- limited to two child nodes per node, which can limit their usefulness in certain apps.
2. Unbalanced trees- one subtree is larger than the other can lead to inefficient search operations.
3. Space inefficiency- each node requires two child pointers which can be a large amount of memory overhead for larger trees.
4. Slow performance- In the worst-case scenario, a binary tree can become degenerate, which means it can degrade to 0(n)-time complexity.
5. Complex algorithms- to balance various algorithms are used that require additional overhead making them not suitable for certain apps.

Vector

Table

Description automatically generated

1(n)

6n+1

1

2

n

n

n

n

n

n

n

n

2

1

1

1

1

Hash Table

Table

Description automatically generated

2

1

2

2

88888 7n+1

0(n)

Binary TreeTable

Description automatically generated

88888 7n+1

0(n)

1 2

1 2

1 1

1 2

**Recommendations**

Based on the Pros and Cons of each Data Structure, I would recommend that the Vector would be best due to its ability to handle large amounts of data and being able to quickly sort to print the catalogue.

References

*Applications, Advantages and Disadvantages of Binary Tree*. (2022, May 25). GeeksforGeeks. https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-binary-tree/

‌

*Applications, Advantages and Disadvantages of Hash Data Structure*. (2022, June 7). GeeksforGeeks. https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-hash-data-structure/

‌

*Understanding Vector Databases: Powering the World of AI*. (n.d.). Www.linkedin.com. Retrieved April 10, 2023, from https://www.linkedin.com/pulse/understanding-vector-databases-powering-world-ai-chris-dougherty

‌