This pseudocode defines how the program opens and reads data from files. It also parses each line for errors.

//Vector Pseudocode initialize a vector called courses.

Struct Course

String courseName;

String courseNumber;

String courseTitle;

String coursePrequisites;

Create vector <Course> theirCourses

Create function to read in file to vector theirCourses

For line in file Check for format error errors

Verify each line has 2 parameters.

Check if each coursePrequisites has matching end matches another line.

If error found return error name.

Create function to create course objects.

Call function to read in file

Call vector to store entries.

Store lines into entries in vector.

Create search function.

Call read file function.

For each item in vector

If item == searchTerm

Print theirCourse

Else

Print Course() empty

//HashTable Pseudocode

Initialize a class for HashTable.

1. Create node and struct course.
2. Add all required variables for use of the class.

Set up constructor and deconstructor.

Create logic for the hash value.

Set up methods to be used on the data.

1. Create course objects method.
   1. Function that can be called to parse data and enter each line into an object.
   2. Data will be loaded into hash table data structure.
2. Display method
   1. Displays current items in hash table
3. Print method
   1. While loop that will print course objects.
   2. Course information will print out.
4. Remove method
   1. That allows user to remove an object from the hash file
5. Search method
   1. Allows user to search for certain course.

//Binary Tree code for computer science

1. Take in data CSV.
   1. Load data
   2. Validate sample file.
      1. Ensure that there are at least 2 parameters on each line.
      2. Ensure prerequisites provided on a line are present at the end of another line.
      3. Ensure the new line starts with he prerequisite course id.
2. Function how to create course object. A while loop.
   1. When data is loaded each line of data will be paced as an object in the binary tree.
   2. The data is ready in and if the prerequisite is required and not present it will be sorted to the left.
   3. If the prerequisite is present, it will be sorted to the right.
   4. These objects will be held within the binary tree.
   5. It will be allowed to be printed.
   6. Once no longer needed deconstructor will remove the data for the current session.

Create function using binary search tree to order the courses in alphabetical order.

1. Call binary tree code.
2. Add in additional step of if the course letters are less than the new addition place it above the course.
3. Else place it below. We will need to have a temp space to sort.
4. I would like to store the data in a vector.
5. This vector will be held and called in the main for printing. PrintMe

Create a print function for Printing courses in computer science in alphabetical order.

Create main

Initialize vector Course theirCourse

Initialize hashtable for use

Create switch cases

1. The person just wants to check the file for errors.
   1. Call the read function.
2. The person just wants to create the objects in the vector to hold.
   1. Call function to create course objects.
3. The person wants to print out specific course information.
   1. Call the search function.
4. Menu that uses case logic that calls upon the already created methods to create new course.

Create menu

1. Option 1 load data structures.
   1. This will allow the person to load in data.
2. Option 2 Print course list.
   1. Will print an ordered list of all courses.
   2. This will use the PrintMe function to print the ordered list.
3. Option 3 Print course
   1. Will print course title and prerequisites for the course.
4. Exit program.

//Inside main adding data for Hash table.

1. Testing-Set up code to open and read in file.
   1. Parse file to validate that two parameters are on each line.
   2. Parse file to validate that any prerequisite provided exists in file.
   3. Check for errors.

Evaluate the run-time and memory of data structures that could be used to address the requirements.

//Vectors

The runtime for opening the file would be O(n) since the file is opened and executed at a cost of 1 per line. Which gives us a Big O value of O(n).

The parsing each line and creating the objects would be a cost per line of 1 and can be executed O(n) times which gives us a Big O value of O(n).

//Hash Table

The runtime for opening the file would be O(n) since the file is opened and executed at a cost of 1 per line. Which gives us a Big O value of O(n).

The parsing each line and creating the objects would be a cost per line of O(1) and can be executed O(n) times which gives us a Big O value of O(n).

//Tree

The runtime for opening the file would be O(n) since the file is opened and executed at a cost of 1 per line. Which gives us a Big O value of O(n).

The parsing each line and creating the objects would be a cost per line of O(log n) and can be executed O(n) times which gives us a Big O value of O(log n).

//Advantages and Disadvantages

The advantages to using Vectors are, they have better performance when it comes to sequential access to data. They can resize easily if needed as well. A disadvantage with vectors is that there is a precision trade off the faster the search the cost of accuracy (Voutila, 2019).

The advantages to using Hash Tables are, hashing typically is the best for large amounts of data storage. “With a good hashing function, hashing is one of the most efficient ways to store, access, and delete data (Chresfield, 2019).” The disadvantage to hashing is, they do not accept null values and require a key which causes collisions.

The advantages of using binary search trees are, you can automatically sort and insert elements. Also, elements are always sorted in some order form. This type of data structure is efficient when it comes to memory. The disadvantages are, they are not suitable for data structures that need to be accessed randomly. They also do not support some functions that are widely used in other data structures. (2023 GeeksforGeeks)

Since the data for reviewing the perquisite and the data for sorting are not linear and vast, I would say there is no need for the binary search tree option which has an O of log n. For this application I will choose the hash table. The data can be linked with key values to capture the links between the prerequisites and that makes the data easily sortable. Also, the big O is minimal for both vector and hash tables therefore the advantages of using hash tipped the scales on this choice. I also feel that coding the hash would be easier as well for insertions and deletions.

Citation

Voutila, D. (2019, October 19). *Vector database vs. graph database in streaming data*. redpanda.com. https://redpanda.com/blog/vector-vs-graph-database-streaming-data

Chresfield, K. (2019, January 21). *Pros & Cons of Hash Tables*. Medium.com. <https://medium.com/@kaelyn.chresfield/pros-cons-of-hash-tables-bc5d3097ffa7>

Unknown. (2023, May 21). *Applications, Advantages and Disadvantages of Binary Search Tree*. geeksforgeeks.org. https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-binary-search-tree/