CSC 307 Data Structures and Algorithm Analysis

Semester Project Requirements

Fall 2025

Overview

Before building a software product/solution, developers need to have a complete understanding of the problem or need that will be solved/fulfilled by the product. Failure to understand the functionality and features needed before development begins on a software product often leads to the project not being complete.

In this phase, groups will determine the necessary features/function that need to be implemented in their semester project solution.

Project Technical Requirements

The project must satisfy the following technical requirements:

1. The primary Data Structure used for the project is of a more complex nature (i.e., a non-linear data structure) is used as the primary means of data management (retrieval, storage, modification, removal). A list of potential data structures is provided in ***Appendix A***.

   Note: Your semester project may utilize additional data structures as a secondary component in the project, which has no restrictions. For example, some algorithms may need a specific data structure to correctly implement it.

2. The data structure MUST support the algorithms necessary for adding, modifying, removing, and organizing data based on the type of data structure being implemented.

   Example: an AVL tree must be able to add new items, look up items, modify items (which may affect their position in the tree) and remove items. Additionally, rotation algorithms must be implemented.

3. The program must support a user interface to command the application. The interface must allow users to execute the data structure operations based on the problem it is solving. Additionally, the application should be user-responsive (i.e., primary commands only carried out when user initiates commands). For example, a command-line menu would solve this.

   Possible C++ GUI frameworks: QT, GTK+ (gtkmm), WX Widgets, Ultimate++, Nana
   Note: The TAs or I will NOT offer troubleshooting for GUI frameworks in C++. You are NOT required to create a GUI application for your project.

4. The program should have some mechanism for storing, loading, and saving data managed by the data structure persistently. (i.e., even if the program exits, data shouldn't be erased). Example mechanisms for persistent storage:
    a. CSV File (a formatted text file that uses commas/endlines to separate data entries)
    b. Local SQL Database (Example: SQLite)
    c. Object Serialization (XML; JSON Storage)

    Note: Teams should not expect the instructor/TA to troubleshoot/debug options other than CSV storage.

5. The program MUST be written in C++.

    Note: The project will receive an automatic zero if the back-end/underlying logic is not written in C++.

Application Commands

Teams should determine the commands that the application should support. This is more directly tied to the problem/need being solved. The following requirements need to be implemented:

1. The application needs some mechanism for allowing the user to add data to the primary data structure.
    a. i.e., insert value, append value, etc.
2. The application needs some mechanism for allowing the user to "read" data stored in a data structure.
    a. i.e., search mechanism, lookup, retrieve/peek next value, etc.
3. The application needs some mechanism for allowing the user the ability to manipulate data stored in a data structure.
    a. i.e., update item data, etc.
4. The application needs some mechanism for allowing the user the ability to remove data stored in the data structure.
    a. i.e., delete item, get next (pop/enqueue), clear data structure, etc.
5. The application needs some mechanism for allowing the user to save data to a CSV file/database/etc.
    a. i.e., a save command.
6. The application needs some mechanism for allowing a user to load data from a CSV file/database/etc.
    a. i.e., load mechanism.
7. The application should have some mechanism for allowing the user to exit the application.
    a. i.e., an exit command when the user is finished using the application.

Submission Instructions

To submit this milestone, students will create a Requirements Document that includes the technical requirement details as well as the application commands. A template for the requirements document will be provided on Canvas alongside this document that the requirements document MUST follow.

Deliverables:

- ***projectname_teamname***_requirements.pdf – a PDF formatted copy of the requirements document.

Rubric and Grading

Criteria	Points Worth
➢ The project was described according to the assignment instructions and project template.	15 pts
➢ The project's technical requirements were described in full according to the assignment instructions and template.	35 pts
➢ The project's application commands were described in full according to the assignment instructions and template.	35 pts
➢ Member contributions were described according to the template.	15 pts
Total	**100 pts**

Penalties	
Template Document Not Followed The project template document was NOT followed.	Up to 100 percent of the project grade depending on severity of feedback. Note that this is an individual penalty and does NOT apply to the entire group.
Late Submissions	There are NO opportunities for late submission of ANY milestones. Any milestone not submitted on time will result in a 0 for that milestone.

Appendix A

The following are potential data structures and their uses to help you best determine the data structure that will help you implement your semester project.

Search Tree Data Structures (balancing trees only):

- Balanced Binary Search Trees:
- AVL
- Red-Black Tree
- Splay Tree
- Trie

Uses:

- Database Indexing – are used to index databases and large data sets to quickly find information based on specific attributes (keywords, tags, user IDs, etc.)
- File Systems – AVL trees may be used to manage file directories.
- Spell Check/Word Match – Data structures like the Trie are efficient at matching typed words. While tries are more tree-like than hash table, they offer similar lookup functionality.

Priority Queueing Data Structures:

- Binary Heap
- Fibonacci Heap

Uses:

- Shortest Path Algorithms in Graphs (Dijkstra's/Prim's) – Shortest path algorithms utilize priority queues to select the node with the smallest known distance from the starting point.
- Data Compression (Huffman Coding) – Used to compress data to take up less space on a system. Huffman Coding Algorithm is used to compress text files or files that contain characters.
- Emergency Room Patient Queuing – Priority Queues may be used to queue up patients at an Emergency Room based on their physical state rather than their time of arrival at the hospital.
- Support/Help Ticketing – Priority Queues may be used to implement priority-based ticket systems found in IT/Support departments.

Indexing/Lookup Data Structures:

- Chained Hash Table
- Probing Hash Table
- Double Hashing

Uses:

- Indexing/Caching – Hash Tables are used to index databases and large data sets to quickly find information based on specific attributes (keywords, tags, user IDs, etc.)
- Sets – Sets are data structures that store unique. Unordered elements. Hash Tables are useful for implementing sets because they may quickly determine if an item already exists in the set.
- Dictionaries (think Python Dictionary) – Dictionaries are data structures that associate a value with a unique key. Hash tables used to implement dictionaries.

Graph Data Structure:

Students may also implement a graph data structure. Graph Data Structures come in many forms, which depend on the problems they are used to solve:

- Directed vs. Undirected (Can traversal between two nodes go both ways for all edges?)
- Weighted vs. Unweighted (Do edges have an associated weight/cost for traversing it?)
- Connected vs. Disconnected (Is it possibles to reach all vertices given any starting vertex?)

Uses:

- Path Navigation – Navigation systems often use graphs to implement pathing between points on a map/area.
- Social Media Modeling – Graphs may help model connections in social media networks. Users are vertices and edges are associations (friends/followers/etc.)

Note that this is not a comprehensive list of data structures and their uses. If you have a particular idea for a program in mind, you are encouraged to research and determine if any non-linear data structures may be used to implement your idea.