CS 2420 - CS 2899 Study Guide

- Understand the basic concept being data containers, including exposed methods, templates, specific targeted purposes of them, etc.
- Know about C++ arrays, how to work with them, how they are managed, and how fast they are in comparison to other collections. How pointers and arrays are two ways of working with the same thing. How to work with pointer arithmetic.
- Explain what an iterator is, and why it is useful.
- Be able to work with linked lists. Know how they work in theory. Know how the code works to add, locate, and delete nodes. Be prepared to write methods for singly linked lists and doubly linked lists.
- Know what a stack is, and common methods to be found for stack objects.
- Know what a queue is, and common methods to be found for a queue.
- Understand Big-O notation. Know what major purposes it is used for. Be able to rank Big-O notations if asked.
- Know how to work with hash tables. How fast can they be? How much memory do they use? What's the differences between an array based and a linked list based hashed table. What's the full process for inserting and retrieving data.
- The basic search algorithms, a sequential search, binary search tree, and hash table searching. Know the Big-O notation of each of these, including average and worst case.
- You should be able to understand how bubble, selection, insertion, quick, merge, and heap sorts work. You should know the average and worst case Big-O notation of each. You should know the Big-O notation of memory usage for these. You should also understand how the bucket/bin sort works. Know what stability means with sorting.
- How to work with binary trees. How to traverse them in an in-order, pre-order, and post-order manner. How to insert and search for items from a binary search tree. How to delete items from a binary tree.
- AVL trees. Why they are useful. How to insert an item into an AVL tree. How AVL tree balance factors work. When an AVL tree will do single and double rotations and what the resulting tree will look like.
- How B Trees work. Why they are useful. What the order of a B Tree means. How nodes are inserted into B Trees. What the resulting tree will look like. What B+ trees are and how they differ from B Trees. Why they are useful.
- Graphs. Various ways to store graphs in a data structure. This includes matrices, adjacency lists, and CSR arrays. How to process data using the Dijkstra's shortest path algorithm. How to traverse a graph using queue/breadth and depth/stack first traversal. What a minimum spanning tree is, and how to use Prim's algorithm to find one.