Data Structures and Algorithms Review

Sorting Algorithms:

- Bubble Sort
    - Sorts by bubbling largest values to the end of the list
    - One approach is to use nested loops
        - Outer loop iteration controls which value you are bubbling up
        - Inner loop iteration operates comparisons between values to find the largest value within that section of the list, swaps values if the left value is greater than the right value
    - Generally utilizes in-place memory swapping
    - Quadratic time complexity
- Selection Sort
    - Sorts by finding the smallest value in a list per iteration and then swapping it into the correct position at the beginning of the list. By doing this you are creating a sorted section at the beginning of the list with unsorted values after it.
    - One approach is to use nested loops
        - Outer loop iteration controls which value you are sorting
        - Inner loop iteration operates comparisons between values to determine if it's the smallest value within that section of the list. Swaps values after having visited every remaining element in the list.
    - Generally utilizes in-place memory swapping
    - Quadratic time complexity
- Merge Sort
    - Divide and conquer sorting algorithm
    - Recursively splits the list in half until sorted single item lists remain and then uses a merge helper function to combine the smaller sorted lists together. As the various recursive invocations of merge_sort pop off the stack the list is merged back together in sorted order.
    - Stable sort(the order of the original elements in the list is preserved)
    - Superlinear time complexity, linear space complexity
- Quick Sort
    - Divide and conquer sorting algorithm
    - Choose a value in the list called the pivot (we normally choose the last value in the list, or the first value in the list).
    - Move all of the values less than the pivot to the left side of the list, and all of the values greater than the pivot to the right side of the list.
    - Put the pivot in between the two.
    - Apply the same sorting function to the partial lists to the left of the pivot and to the right of the pivot.
    - Basically each time we end up choosing a pivot value and then running partition to determine it’s position we are putting that element in it’s correct position were the list to be sorted

    - Unstable sort(the order of the original elements in the list is NOT preserved)
    - Superlinear time complexity, Logarithmic space complexity

Data Structures:

- Graphs
    - A data structure that stores a collection of node structures in non-contiguous memory.
    - Can be represented 3 ways:
        - Links and Nodes
            - A node contains a value along with a collection of links(references) to related nodes
        - Adjacency Matrix
            - Instead of representing a graph via nodes and links we can represent a graph via a matrix where each node is represented by a column and row in the matrix.
            - If there is a connection between nodes, we represent it as a 1 in the matrix.
            - If there isn't a connection between nodes, we represent it as a 0 in the matrix
        - Adjacency List
            - We represent the graph as a dictionary where each key, the outgoing edge, represents a node in the list. The values associated with that key, incoming edges, are a list of related nodes.
    - The different representations of a graph give us the ability to apply different types of algorithms that are optimized for particular configurations.
    - A directed graph refers to a graph where you can only move from one node to another in a predetermined pattern.
    - A cycle is when following a path through the nodes causing you to return to a node you have already visited in a graph
- Linked List
    - A data structure that stores a collection of node structures in non-contiguous memory. The Linked List structure itself generally consists of a head and tail reference which are references to the first and last element in the linked list.
    - A node generally consists of a value and next reference, the reference to the next node in the linked list. A variant of a singly linked list, the doubly linked list, adds an additional previous reference to each node, the reference to the previous node in the list.
    - SLL:
        - Linear access or search, but generally considered to have constant time insertion and removal
    - DLL:
        - Linear access or search, but generally considered to have constant time insertion and removal

    - It is a special type of graph
- Trees
    - A type of directed graph designed to represent hierarchical relationships between a collection of nodes.
    - Each node in the tree generally consists of a reference to a value along with a reference to a collection of children.
    - Trees don’t have cycles
- Breadth First Search vs Depth First Search
    - Algorithms used to perform traversals on graph type structures
    - Breadth First Search
        - In Breadth First Search, the algorithm starts at the root node of the graph/tree and then visits every child on each level of the graph before moving onto subsequent levels.
        - To do this, BFS generally uses a queue data structure to keep track of the nodes left to visit.
            - Queues are FIFO(First In/First Out)
        - Can be implemented with a while loop or recursively
        - Pseudocode
            - First, check if the root node is the target. If it is, return the root node.
            - Otherwise, add the children of the current node to the queue of nodes.
            - Pop each from the queue and check if each are the target.
            - Return if target. Otherwise, go to step 2 and repeat.
        - Good for finding values that are close to the root node
    - Depth First Search
        - In Depth First Search, the algorithm starts at the root node of the graph/tree and then, generally, goes all the way down the leftmost branch of the tree looking for certain terminating conditions such as a node with a specific value or a node with no children. Once we have found a node with no children, the algorithm then checks the right side of the parent node. This process continues until either a target value is found or you have visited every node in a graph/tree.
        - To do this, DFS generally uses a stack data structure to keep track of the nodes left to visit.
            - Stacks are LIFO(Last In/First Out)
        - Can be implemented with a while loop or recursively
        - Pseudocode of Depth First Search
            - First, check if the root node is the target. If it is, return the root node.
            - Otherwise, add the children of the current node to the stack of nodes.
            - Pop each from the stack and check if each are the target.
            - Return if target. Otherwise, go to step 2 and repeat.

        - Good for finding values that are far away from the root node
- Binary Search and Binary Search Trees
    - Binary Search
        - A search algorithm that can be used on sorted data sets that cuts the search area in half with each successive binary search operation.
        - Logarithmic time complexity.
    - Binary Search Tree
        - A special type of tree, search optimized data structure, where the nodes in the tree follow a specific pattern. Left child nodes are smaller than the parent node. Right child nodes are larger than the parent node.
        - Because of this orientation, you can apply a binary search algorithm to generally find nodes in logarithmic time(average case).
            - This average case time complexity of logarithmic time depends on the relationship between the length of the leaves and the number of nodes in the tree.
        - This orientation has some downsides though. It affects insert operations as additional nodes added to a BST may require rebalancing of branches or potentially the entire tree.