**Stacks and Queues : Dequeue**/ remove from the front of a queue. **Enqueue**/ Remove from the back of a queue.

If you hit the back of a queue and want to enqueuer, the enqueuer would be at the front of the queue and be a circler buffer.

**Deep copy**/ copies all the information including ptr addresses. **Shallow Copy**/ copies all the information Expect ptr addresses

**Queue/** is a FIFO data structure, like a line of people. **Peek/** retrieve a copy of the element at the front of the line without removing it.

**Peek/** means retrieve the top of the stack without removing it.

**Pop**/ means take a data element off the top of the stack **Push/** means place a new data element at the top of the stack. **Stack/** the set of data is the stack of elements.

**Time Complexities/** Are used to help make an intelligent decision about which algorithm to use, when two algorithms accomplish the same task. **Cross-over point/** The intersection of two graph’s algorithms with different time complexities. **Asymptotic running time/** running time of an algorithm as the number of elements approaches infinity. **Problem Size/** The amount of the time an algorithm takes to execute is represents by the number of instructions that are executed. **Determining a Time Complexity/** if a function for the number of instructions has at least one term with n, we can determine the time complexity by: 1. removing the least significant terms from the function 2.Removing the coefficient of the remaining term.

**Binary Search Trees/** Is a binary tree that allows us to search for values that can be anywhere in the tree. Time for insertion is theta(log n) Time for search is O(log n) **Balanced binary search tree/** if a binary search tree takes theata(log n ) for Insertion and O(log n ) for searching. **Un-Balanced binary search trees /** if a binary search tree takes O(n) time for insertions and searchings. Binary search trees are faster on average than red-black/ AVL trees, however red-black / AVL trees give consistent performance.

**Trees :** a heap is a type of tree **Tree/** Is a set of linked nodes , such that there is one and only one path from a unique node (called the root node) to every other node in a tree. There are no cycles in trees. **Subtree/**  is a part of a tree that is a tree itself. **Binary Tree/** is a tree in which each node can only have up to two children. The links in a tree are often called edges. **Full Binary Tree/** each node has two children except for the nodes on the last level, which are leaf nodes, is also called a complete binary tree. **Complete Binary Tree/** is a binary tree that is either: 1 a full binary tree. 2: a tree that would be a full binary tree but it is missing the rightmost nodes on the last level. **Root node/** The node at the top of the tree that has no parents. **Leaf nodes/** nodes that have no children **Child node or Children/** Nodes below a given node **infix or inorder /**  Traverse left subtree of node, Process data in node, Traverse right subtree of node **Preorder or Prefix /** Process data in node, Traverse left subtree of node, Traverse right subtree of node. **Postorder or Postfix /** Traverse left subtree of node, Traverse right subtree of node, Process date in node.

**Big Oh /** Summary Big-O Notation, where *n* refers to the size of the problem (e.g., n is the length of the array) …O(1) = “Constant Time” – runtime does not depend on n ….O(log n) = “Logarithmic Time” – runtime is proportional to log n ….Clue: Every time you double the problem size, runtime grows by a constant …O(n) = “Linear Time” – runtime is proportional to n ….Clue: Every time you double the problem size, time doubles …..O(n^2) = “Quadratic Time” – runtime is proportional n^2 ….Clue: A linear time operation applied a linear number of times …O(2^n) = “Exponential Time” – runtime is proportional 2^n …Clue: Add one to the problem size, runtime doubles

A pointeris a variable used to store an address. Referencing operator = &, an address….Differencing operator = \*,location at the address… When a program uses memory from the heap, the used heap memory is called dynamically-allocated memory