++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

DS

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

1. Stack, Arrays, Queue, Circular Queue( rear = (rear + 1)% length )
2. Singly, doubly, circular, node
3. Recursion - function calls itself
   1. Direct - function calls the same function
   2. Indirect - function calls other function and this function calls first function
4. Complexity -
   1. Two functions - 2^n
   2. One Function - n
   3. Function and for loop - n^2
   4. Function(n/5) - logn
5. Tree

* Terminology - root, edge, parent, child, sibling, leaf nodes, level, height
* Traversal - inorder (l, R, r), preorder (root, left, right), inorder (left, right, root)
* Types
  + Binary tree - zero, one, two childs
  + Complete binary tree - all levels are completely filled except lowest level (almost complete tree )
  + Full binary tree ( perfect )
  + Binary search tree - left and right are in ascending
  + AVL - height balanced tree
  + Multi way tree - can have more no of childs
  + B-tree - balanced tree logn complexity
  + B+tree -
  + BFS / DFS

1. Searching

* Binary search logn

1. Sorting
   * Selection - element is picked and put at position n^2
   * Insertion - position is picked and put the value n^2
   * Bubble - two neighbour element is compared n^2
   * Merge - array is halved and then after sorting merged nlogn
   * Quick - pivot element is selected and swapping is done nlogn
   * Heap - -------------------------
2. Hashing
   * Collision

Linear probing

Quadratic probing

Double probing

Open Hashing - separate chaining

Closed Hashing - open addressing

1. Graphs

* Terminology
  + Path, closed path, simple path, cycle, connected graph, weighted graph, directed graph, adjacent nodes, degree of node (isolated node - degree 0)
* Types
  + Non directed, directed, connected,
  + Complete - all vertices are connected
* Representation
  + Adjacency matrix - n \* n (no of vertices) shows edges of vertices (2 d array)
  + Adjacency list - array of linked list size same as number of vertices
  + Traversal - BFS ( queue ) , DFS ( stack )
* Shortest Path
  + Dijkstra - greedy (n^2) level setting
  + Floyd-Warshall - all pair shortest path level correcting dp
* Spanning tree it is subset of graph does not have cycle it is not disconnected and it is uni-directed
  + - Minimum spanning tree
  + Kruskal - starts from any vertices, many node at a time, only connected, for dense graph, (v-1) edges weighted greedy
  + Prims - start with empty tree, only node at a time, disconnected, for sparse graphs, maintains two set of vertices first is already exists and other does not exists picks edges and then moves to other endpoint greedy

1. Algorithms
   * Brute force
   * Divide and Conquer - quick, merge, closed pair, strassens, cooley, Karatsuba
   * Greedy - choose next most obvious answer fractional knapsack
   * D P - stores sub problems do not re compute
   * Back tracking - removing solutions from recursion when fails to satisfy conditions n queens, knapsack, hamiltion, generate k array, graph coloring
   * Branch and bound - combinational, exponential complexity,
   * Stochastic