Ch-1- Introduction to Stacks, Queues and Linked list

 # STACKS:
- → defination ⟶ Push, POP, Peer, isempty, isfull
- → using array
- → using linked list
- → Appⁿ
  - → infix to postfix
  - → infix to prefix   } defination + Algorithm.
  - → postfix to infix
  - → prefix to infix

 # Queues
- → defination
- → Queue as an ADT
  - operations ⟶ enqueue
    - ⟶ dequeue.
- → using array.
- → using linked list.
- → circular Queue [defi + Algo].
- → Appⁿ of Queue [Scheduling].
- → Priority Queue
- → Double ended Queue
  - → defi
  - → Algo
  - → Operation
  - → as an ADT

 # Linked list -
- → what is linked list
- → Advantages + disadvantages
- → diff between array, linked list
- → Types
  - → Singly linked list
    - → create
    - → insert (all cases)
    - → delete
    - → traverse.
  - → doubly insert (all cases)
    - → Advantage
    - → create
    - → traversal
    - → display

Circular Linked list

- definition
- operations → Insert node
  - traverse
  - delete
  - display

→ Reversing singly linked list

→ App^n → Adding 2 polynomials
  → multiplication of 2 polynomials.

★ → Extra important Questions

1. ADT
2. Primitive and non-primitive
3. Linear and Nonlinear
4. Static and Dynamic
★ 5. Asymptotic Notation.
6. Differentiate between space complexity and time complexity.

Ch-2 - Trees.

 # Basics
- root
- parent
- child
- siblings
- path
- Degree of tree, Height

 # Binary Tree → defination
→ function (algo)

 # General Tree → defination
→ example.

 # Types of Binary Tree
- full binary Tree
- complete binary Tree
- Strictly Binary Tree
- Extended Binary Tree.

 # Tree Traversal
- inorder
- Preorder
- postorder

[defination + function]

 # Operation in BST
- insertion
- deletion.
- search.

 # AVL Tree
→ defination.
→ insertion (function)
→ Deletion (function)

 # Application of Tree
- expression Trees. ⎫
- Huffman Tree ⎬ defination, algo,
- ★ Heap Sort + Algo ⎭ complexity.
- B Tree
- B+ Tree
- Trie indexing.

 # Numerical
→ construst BST
→ find inorder preorder postorder.
→ construct AVL
→ construct Tree [if inorder, preorder are given]

Ch-3 — <u>Graphs</u>.

| Basic |
→ Graph ?
→ undirected graph
→ directed graph
→ complete graph
→ weighted graph
→ Adjacent nodes
→ path
→ Cycle
→ connected graph.
→ Subgraph.

→ Degree of vertex
→ self edges, loops
→ multigraph.
→ Tree
→ Spanning Tree
→ minimum Spanning Tree.

| Representation |
→ Adjacency matrix
→ Adjacency list
→ path matrix

| Traversals |
→ BFS   [algorithm, definition
→ DFS   [- example]

 # ? | Application of Graph |

| Algorithms |
→ Prism
→ Kruskals
→ Dijkstras

| Numerical : |

① find MST with cost using all algorithm.
② find ajacency list and matrix.

\# [Recursion]
  → defination.
  → advantages
  → disadwantages.

→ [winding and unwinding phase]

→ [Tail Recursion.] [+example]
→ [Direct Recursion]
→ [Indirect Recursion].
→ [Tree Recursion] (eg fibonacii)

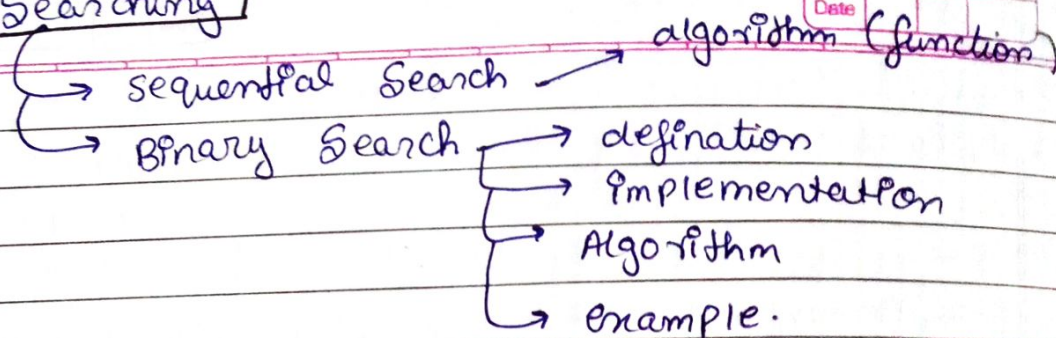→ Difference between Recursive and iterative.

\# [Storage management]
  → defination
  → firstfit   } defination +
  → bestfit   numerical.
  → worse fit methods.

  → fragmentation
  → freeing memory.

  → Binary Buddy System
  → fibonacci Buddy System.

\# compaction and [Garbage Collection].
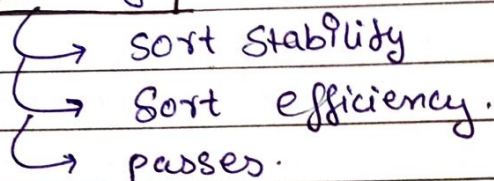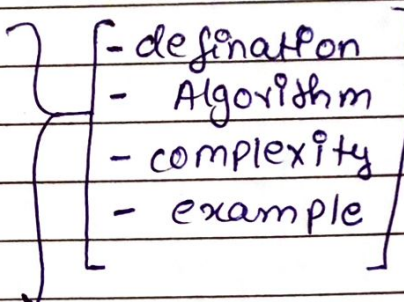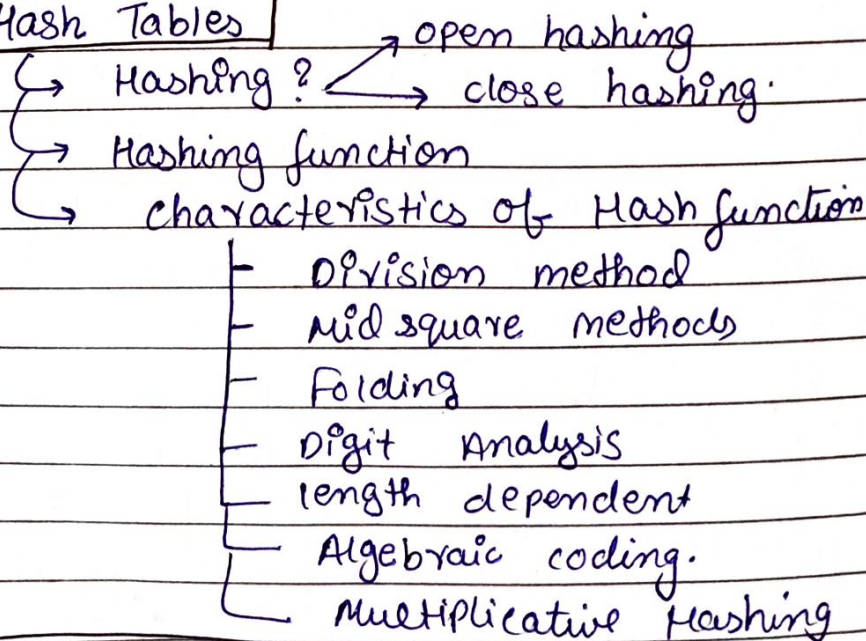
Ch-5 → Searching and Sorting.

| Searching |
- → sequential Search → algorithm (function)
- → Binary Search → defination
  - → implementation
  - → Algorithm
  - → example.

| Sorting. |
- → sort Stability
- → Sort efficiency.
- → passes.

#
1. Insertion Sort
2. Selection Sort
3. Merge Sort
4. Quick sort.
5. Radix Sort.
6.

  - defination
  - Algorithm
  - complexity
  - example

| Hash Tables |
- → Hashing? → open hashing
  - → close hashing.
- → Hashing function
- → characteristics of Hash function
  - Division method
  - Mid square methods
  - Folding
  - Digit Analysis
  - length dependent
  - Algebraic coding.
  - Multiplicative Hashing

| Collision Resolution | → Separate charning
- → open addressing → linear prob
  - → quadratic pro
- → double hashing

- Bucket hashing.