DS- WEEK 3

Tree Concepts

- **Definition and Properties of Trees**
- **Types of Trees**
  - Complete Tree
  - Full Tree
  - Perfect Tree
- **Node Properties**
  - Height vs. Depth of a Node
  - Degree of a Node
- **Applications of Trees**

Binary Search Tree (BST) Concepts

- **Definition and Properties of BST**
- **Operations on BST**
  - Insertion
  - Deletion
  - Search (contains)
- **Tree Traversals**
  - In-order Traversal
  - Pre-order Traversal
  - Post-order Traversal
- **Applications of BST**
- **Advanced BST Concepts**
  - Validating a BST
  - Finding the Closest Value to a Given Number in a BST
  - Self-Balancing Trees (AVL Trees, Red-Black Trees)
  - Checking if a BST is a Subset of Another Tree

Heap Concepts

- **Definition and Properties of Heaps**
- **Types of Heaps**
  - Max-Heap
  - Min-Heap
- **Operations on Heaps**
  - Insertion
  - Deletion
  - Peek (get max/min)
- **Heapification Methods**
  - Top-Down Heapification
  - Bottom-Up Heapification
- **Advanced Heap Concepts**

    - Binomial Heap
- **Applications of Heaps**
    - Priority Queues

Heap Sort

- **Concept and Algorithm**
- **Time Complexity: O(n log n)**
- **Applications of Heap Sort**

Trie Concepts

- **Definition and Properties of Tries**
- **Operations on Tries**
    - Insertion
    - Deletion
    - Search
- **Applications of Tries**
    - Autocomplete
    - Spell Checker
- **Comparison with Hash Tables**

Graph Concepts

- **Definition and Properties of Graphs**
- **Types of Graphs**
    - Directed vs. Undirected Graphs
    - Weighted vs. Unweighted Graphs
- **Special Types of Graphs**
    - Directed Acyclic Graph (DAG)
    - Cyclic Graphs
    - Isolated Vertex
- **Graph Representations**
    - Adjacency List
    - Adjacency Matrix
- **Applications of Graphs**

Graph Traversals

- **Breadth-First Search (BFS)**
    - Concept and Algorithm
    - Applications
- **Depth-First Search (DFS)**
    - Concept and Algorithm
    - Applications

Advanced Graph Concepts

- **Graph Indexing**
  - Vertex Indexing
  - Edge Indexing
- **Shortest Path Algorithms**
  - Dijkstra's Algorithm
  - Bellman-Ford Algorithm

Additional Topics

- **Priority Queues Using Heaps**
- **Trie vs. Hash Table**
- **Self-Balancing Trees**
  - AVL Trees
  - Red-Black Trees
  - Splay Trees
  - B-Trees

DS- WEEK 3 : PRACTICALS

Tree Concepts

1. **Tree Operations**
   - Implement a basic tree structure.
   - Write a function to calculate the height of a tree.
   - Write a function to calculate the depth of a node.
   - Write a function to find the degree of a node.
2. **Tree Traversals**
   - Implement in-order traversal.
   - Implement pre-order traversal.
   - Implement post-order traversal.
3. **Tree Applications**
   - Write a function to check if a tree is complete.
   - Write a function to check if a tree is full.
   - Write a function to check if a tree is perfect.

Binary Search Tree (BST) Concepts

1. **BST Operations**
   - Implement a BST with insertion, deletion, and search (contains) operations.
2. **Tree Traversals in BST**
   - Implement in-order traversal for a BST.
   - Implement pre-order traversal for a BST.
   - Implement post-order traversal for a BST.
3. **BST Applications**

- Write a function to find the closest value to a given number in a BST.
- Write a function to validate if a tree is a BST.
- Implement self-balancing BSTs (AVL Tree, Red-Black Tree).
- Write a function to check if a BST is a subset of another tree.

Heap Concepts

1. **Heap Operations**
   - Implement a max-heap with insertion, deletion, and peek operations.
   - Implement a min-heap with insertion, deletion, and peek operations.
2. **Heapification Methods**
   - Implement top-down heapification.
   - Implement bottom-up heapification.
3. **Heap Applications**
   - Write a function to implement a priority queue using a heap.

Heap Sort

1. **Heap Sort Implementation**
   - Implement the heap sort algorithm.
   - Write a function to sort an array using heap sort.

Trie Concepts

1. **Trie Operations**
   - Implement a trie with insertion, deletion, and search operations.
2. **Trie Applications**
   - Write a function for autocomplete using a trie.
   - Write a function for spell checking using a trie.

Graph Concepts

1. **Graph Representations**
   - Implement a graph using an adjacency list.
   - Implement a graph using an adjacency matrix.
2. **Graph Operations**
   - Write functions to add and remove vertices and edges.
3. **Special Graph Types**
   - Write a function to check if a graph is a Directed Acyclic Graph (DAG).
   - Write a function to find isolated vertices.

Graph Traversals

1. **Breadth-First Search (BFS)**
   - Implement BFS for a graph.
   - Write a function to find the shortest path using BFS.
2. **Depth-First Search (DFS)**
   - Implement DFS for a graph.

    - Write a function to find connected components using DFS.

Advanced Graph Concepts

1. **Shortest Path Algorithms**
    - Implement Dijkstra's algorithm.
    - Implement Bellman-Ford algorithm.
2. **Graph Indexing**
    - Write functions for vertex indexing and edge indexing.

Additional Topics

1. **Priority Queues Using Heaps**
    - Write a function to implement a priority queue using a heap.
2. **Trie vs. Hash Table**
    - Write a comparison of trie and hash table for different use cases.
3. **Self-Balancing Trees**
    - Implement an AVL tree.
    - Implement a Red-Black tree.
    - Implement a Splay tree.
    - Implement a B-tree.