## Main topics:

#### **Tree Concepts**

* Tree Terminology: Root, Parent, Child, Leaf, Depth, Height
* Implement a basic Tree structure
* Tree Traversals: Preorder, Inorder, Postorder
* Common Tree operations (Insertion, Deletion, Searching)

#### **Binary Search Tree (BST) Concepts**

* Properties of BST
* Implementing BST with:
  + **Insertion**
  + **Search (Contains function)**
  + **Deletion** (Handling different cases: leaf node, one child, two children)
* Tree Traversals: Inorder, Preorder, Postorder
* Find the closest value to a given number in BST
* Validate if a given tree is a BST

#### **Heap Concepts**

* Understanding Min Heap & Max Heap
* Heap Operations:
  + **Build Heap**
  + **Insert**
  + **Remove (Extract Min/Max)**
* Heap Sort Algorithm

#### **Trie Concepts**

* Understanding Trie (Prefix Tree)
* Implement Trie with:
  + **Insert words**
  + **Search for a word**
  + **Delete words**
* Applications of Trie (Autocomplete, Spell Checking)

#### **Graph Concepts**

* Graph Representations:
  + **Adjacency Matrix**
  + **Adjacency List**
* Graph Traversals:
  + **Breadth-First Search (BFS)**
  + **Depth-First Search (DFS)**
* Detecting Cycles in a Graph
* Shortest Path Algorithms (Dijkstra’s, Bellman-Ford)

#### **Practice & Applications**

* Solve **at least 3 problems** for each data structure from competitive coding platforms
* Explore real-world applications of each data structure (e.g., Heaps in priority queues, Trie in search engines, Graphs in social networks)

**📌 Week 10: Data Structures & Algorithms 3**

**1️⃣ Trees**

* **Concepts of Trees**
  + Tree Terminologies: Root, Parent, Child, Leaf, Depth, Height, Degree, etc.
  + Types of Trees: Binary Tree, BST, AVL Tree, Red-Black Tree, Full Binary Tree, Complete Binary Tree
  + Tree Traversals: Preorder, Inorder, Postorder, Level-order
  + Binary Tree vs. Binary Search Tree (BST)
  + Implement a **Binary Tree** (not BST)
  + Check if a Tree is Balanced (isBalanced Tree)
  + Find **Depth & Height** of a node
  + Find **LCA (Lowest Common Ancestor)** of two nodes
  + Count **Single Child Nodes** in a BST
* **Sample Workouts (Practice Questions)**
  + Implement Tree Traversals (Preorder, Inorder, Postorder, Level-order)
  + Implement a Binary Tree and Insert Nodes
  + Check if a given Tree is Balanced

**2️⃣ Binary Search Tree (BST)**

* **Concepts of BST**
  + **BST Operations:** Insert, Search, Delete
  + **Tree Validations:**
    - Check if a given tree is a **valid BST**
    - Find the **closest value** to a given number in BST
  + **Kth Element Problems**
    - Find the **Kth Smallest Element** in BST
    - Find the **Kth Largest Element** from an array
  + **Applications of BST**
    - Auto-suggestions
    - Indexing in databases
* **Sample Workouts (Practice Questions)**
  + Implement a BST with Insert, Search, and Delete
  + Validate if a Tree is BST
  + Find Closest Value in BST

**3️⃣ Heap**

* **Concepts of Heap**
  + **Types of Heaps:**
    - **Min-Heap** vs **Max-Heap**
  + **Operations in Heap**
    - **Build a Heap** (Heapify)
    - **Insert** a new element
    - **Remove** an element (Extract Min/Max)
  + **Priority Queues using Heaps**
  + **Heap Applications:**
    - Scheduling tasks
    - Finding top K frequent elements
* **Sample Workouts (Practice Questions)**
  + Implement Min-Heap & Max-Heap
  + Build a Heap using Heapify
  + Insert and Remove from a Heap

**4️⃣ Heap Sort**

* **Concepts of Heap Sort**
  + **Implementation of Heap Sort**
  + **Complexity Analysis**
  + **Comparison with Merge Sort & Quick Sort**
  + **When to Use Heap Sort?**
  + **Limitations of Heap Sort**
* **Sample Workouts (Practice Questions)**
  + Implement Heap Sort
  + Analyze Time Complexity
  + Sort an Array using Heap Sort

**5️⃣ Trie (Prefix Tree)**

* **Concepts of Trie**
  + **Trie Operations:**
    - Insert
    - Search
    - Delete
  + **Applications of Trie:**
    - **Auto-completion**
    - **Spell Checking**
    - **Longest Common Prefix**
  + **Prefix Trie vs Suffix Trie**
  + **Suffix Tree Applications**
* **Sample Workouts (Practice Questions)**
  + Implement Trie with Insert, Search, Delete
  + Auto-suggestion using Trie
  + Find Longest Prefix in a given set of words

**6️⃣ Graph**

* **Concepts of Graphs**
  + Graph Terminologies: Vertex, Edge, Degree of a Node
  + Types of Graphs:
    - **Directed vs Undirected Graph**
    - **Weighted vs Unweighted Graph**
    - **Complete Graph vs Disconnected Graph**
  + **Graph Representations:**
    - **Adjacency Matrix vs Adjacency List**
  + **Graph Operations & Complexity:**
    - BFS Complexity
    - DFS Complexity
    - Time Complexity of Graph Operations
* **Sample Workouts (Practice Questions)**
  + Represent a Graph in memory using **Adjacency List**
  + Find the **Degree of a Node** in a Graph
  + Clone a Graph (Leetcode Question)

**7️⃣ Graph Traversals (BFS & DFS)**

* **Concepts of BFS & DFS**
  + **BFS (Breadth-First Search)**
  + **DFS (Depth-First Search)**
  + **Applications of Graph Traversals:**
    - Finding **Shortest Path** in an Unweighted Graph
    - **Cycle Detection** in a Graph
    - **Connected Components**
* **Sample Workouts (Practice Questions)**
  + Implement **BFS & DFS** in a Graph
  + Find **Shortest Path** using BFS
  + Detect **Cycles** using DFS

**8️⃣ Graph Algorithms**

* **Minimum Spanning Tree (MST)**
  + **Kruskal’s Algorithm**
  + **Prim’s Algorithm**
* **Shortest Path Algorithms**
  + **Dijkstra’s Algorithm**
  + **Bellman-Ford Algorithm**
* **Applications of Graph Algorithms**
  + Network Routing
  + Social Media Friend Suggestions
* **Sample Workouts (Practice Questions)**
  + Implement **MST using Kruskal’s Algorithm**
  + Implement **Dijkstra’s Algorithm**
  + Find **Shortest Path between 2 Vertices**

**📝 Additional Topics (Pending)**

* **Binary Tree vs BST - Difference in Search Time Complexity**
* **Find the Second Largest Element in BST**
* **Implement Balanced Tree**
* **Quadratic vs Linear Time Complexity**
* **Graph Indexing**
* **Spanning Tree & Its Uses**
* **Applications of Trees, Graphs, Heaps & Tries**
* **Event Propagation & Delegation (JS-related topics, but good to know)**
* **Redux Thunk & Fiber (React-specific topics)**

**📌 Practice Plan**

✅ **Competitive Coding Practice**

* Solve at least **3 problems per data structure** from:
  + **Leetcode (Blind 75 List)**
  + **GeeksforGeeks**
  + **HackerRank**
* Learn **Optimal Solutions from YouTube** after solving problems

**🚀 Key Focus Areas**

✔️ **Master Tree, Heap, Trie & Graphs**  
✔️ **Implement All Core Data Structures from Scratch**  
✔️ **Understand Time Complexity of Each Operation**  
✔️ **Solve Problems from Leetcode & Learn Optimal Solutions**  
✔️ **Understand Real-World Applications of Each Data Structure**