Pending's PDF: DSA-3 Topics Compiled

by Fahad VM

1 Theory

- **Trees:**
  - Complete tree vs. full tree
  - Perfect tree vs. full tree
  - Balanced tree vs. unbalanced tree
  - Binary tree vs. binary search tree (BST)
  - Applications of trees
  - Applications of BST
  - BST vs. heap
  - N-ary tree
  - Ternary tree
  - Degenerate tree
  - Height of a node vs. depth of a node
  - Degree of node vs. degree of tree
  - Internal nodes
  - Siblings
  - Terminologies in trees (e.g., leaf nodes, root, parent, child)
  - AVL tree concepts
  - Red-black tree concepts
  - B-tree concepts
  - Segment tree concepts
  - Time complexity of search, insert, delete in binary tree vs. BST
  - Time complexity of BST insertion
  - Complexity of removing second largest element in BST
  - Tree vs. graph
- **Graphs:**
  - Types of graphs (directed vs. undirected, weighted vs. unweighted, complete, disconnected, bipartite)
  - Applications of graphs (e.g., social media mutual friends, maps)
  - Adjacency list vs. adjacency matrix
  - Complexity of BFS and DFS in graphs
  - Complexity of graph initialization
  - Minimum spanning tree concepts
  - Spanning tree concepts
  - Use of spanning tree
  - How to detect cycles in a graph

  - How to count cycles in a graph
  - Shortest path in a graph (weighted and unweighted)
  - Dijkstras algorithm
  - Prims algorithm
  - Kruskals algorithm
  - Degree of vertex
  - Backtracking in DFS
  - Graph indexing
  - Representing a graph in memory
  - Applications of weighted graphs
  - Loops in graphs

- **Tries:**
  - Concept of trie
  - Types of tries (suffix vs. prefix, compressed trie)
  - Advantages of tries
  - Applications of tries (e.g., auto-completion, word suggestion)
  - Complexity of trie operations (initialization, insert, search)
  - Suffix trie vs. prefix trie
  - Trie serialization and deserialization

- **Heaps:**
  - What is a heap?
  - Min heap vs. max heap
  - Is heap a complete binary tree?
  - Applications of heaps
  - Priority queue and heap
  - Applications of priority queue
  - Complexity of heap sort
  - Complexity of heap operations (insert, delete)
  - Heapify (up and down) concepts
  - Limitations of heaps

- **Complexity Analysis:**
  - Quadratic time complexity
  - Linear time complexity
  - Complexity analysis of various algorithms
  - Time complexity of graph operations

  - Time complexity of tree operations
  - Logarithmic values and functions
  - Complexity of BFS and DFS
  - How to find complexities of various algorithms

- **Miscellaneous:**
  - Linear vs. non-linear data structures
  - Advantages of recursion
  - Applications of hash tables

2 Practical

- **Tree Operations:**
  - Implement a binary tree (not BST)
  - Implement a binary search tree (BST)
  - Implement a tree with multiple children per node
  - Implement a balanced binary tree (not BST)
  - BST insertion
  - BST deletion (including special cases)
  - Validate if a tree is a BST
  - Check if a tree is balanced
  - Find height of a BST
  - Find the second largest element in a BST
  - Find the third largest element in a BST
  - Find the kth smallest element in a BST (using inorder traversal, LeetCode 230)
  - Find the kth largest element in a BST
  - Find the element closest to a target value in a BST
  - Check if two BSTs are identical
  - Check if one tree is a subtree of another
  - Level order traversal of a binary tree (LeetCode 102)
  - Postorder traversal
  - Preorder traversal
  - Print all leaf nodes in a tree
  - Find the lowest common ancestor (LCA) in a BST
  - Count single child nodes in a BST
  - Allow duplicate elements in a BST

  - Find min in BST using recursion

- **Graph Operations:**
  - Implement a graph (using adjacency list or matrix)
  - Graph traversal using BFS
  - Graph traversal using DFS
  - Clone graph (LeetCode 133)
  - Number of islands (LeetCode 200)
  - Find shortest path in an unweighted graph (BFS)
  - Find shortest path in a weighted graph (Dijkstras algorithm)
  - Find shortest distance between two vertices
  - Check cycles in a graph
  - Count cycles in a graph

- **Trie Operations:**
  - Implement a trie
  - Auto-completion using trie
  - Word suggestion using trie
  - Search a word in a trie
  - Prefix search in a trie
  - Find the longest prefix in a trie
  - Insert a new word into a trie
  - Deletion in a trie
  - Find the longest non-repeating substring in a string

- **Heap Operations:**
  - Implement a min heap
  - Implement a max heap
  - Heap sort implementation
  - Max heap sort implementation using heapify
  - Min heap insertion
  - Delete node from a heap
  - Delete node from a min heap
  - Convert min heap to max heap
  - Find the right child of a heap
  - Top K frequent elements using heap
  - Kth largest element in an array using heap

- **Array/String Operations:**

  - Find combination of numbers that sum to 2 (e.g., [1,2,-3,5,0,4,-8,-5,7,-9,-7,3])
  - Find unique characters from a given string

- **General Problem-Solving:**
  - Practice problems from Blind 75 LeetCode (learn brute force and optimal solutions from YouTube)
  - Solve LeetCode 215 (Kth Largest Element in an Array)
  - Improve problem-solving, logic, and coding speed (complete complex problems in 5 minutes)
  - Learn to read errors and debug code
  - Practice logical workouts
  - Solve problems on LeetCode, HackerRank, and GeeksforGeeks

- **Sliding Window:**
  - Apply sliding window pattern

3 Results

- Placeholder for results (to be updated with specific outcomes or progress).