**Binary Tree:**

|  |  |  |
| --- | --- | --- |
|  | Problem | Status |
|  | Binary Tree | Set 1 (Introduction) |  |
|  | Binary Tree | Set 2 (Properties) |  |
|  | Binary Tree | Set 3 (Types of Binary Tree) |  |
|  | Handshaking Lemma and Interesting Tree Properties |  |
|  | Enumeration of Binary Trees |  |
|  | Applications of tree data structure |  |
|  | BFS vs DFS for Binary Tree |  |
|  | Tree Traversals |  |
|  | Level Order Tree Traversal |  |
|  | Print level order traversal line by line | Set 1 |  |
|  | Inorder Tree Traversal without Recursion |  |
|  | Inorder Tree Traversal without recursion and without stack! |  |
|  | Iterative Preorder Traversal |  |
|  | Morris traversal for Preorder |  |
|  | Iterative Postorder Traversal | Set 1 (Using Two Stacks) |  |
|  | Iterative Postorder Traversal | Set 2 (Using One Stack) |  |
|  | Reverse Level Order Traversal |  |
|  | Print Postorder traversal from given Inorder and Preorder traversals |  |
|  | Level order traversal line by line | Set 2 (Using Two Queues) |  |
|  | Diagonal Traversal of Binary Tree |  |
|  | Inorder Non-threaded Binary Tree Traversal without Recursion or Stack |  |
|  | Check if leaf traversal of two Binary Trees is same? |  |
|  | Print a Binary Tree in Vertical Order | Set 1 |  |
|  | Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method) |  |
|  | Boundary Traversal of binary tree |  |
|  | Perfect Binary Tree Specific Level Order Traversal |  |
|  | Perfect Binary Tree Specific Level Order Traversal | Set 2 |  |
|  | If you are given two traversal sequences, can you construct the binary tree? |  |
|  | Construct Tree from given Inorder and Preorder traversals |  |
|  | Construct a tree from Inorder and Level order traversals |  |
|  | Construct Complete Binary Tree from its Linked List Representation |  |
|  | Construct Full Binary Tree from given preorder and postorder traversals |  |
|  | Construct a special tree from given preorder traversal |  |
|  | Construct tree from ancestor matrix |  |
|  | Construct Ancestor Matrix from a Given Binary Tree |  |
|  | Construct Special Binary Tree from given Inorder traversal |  |
|  | Construct Binary Tree from given Parent Array representation |  |
|  | Construct a Binary Tree from Postorder and Inorder |  |
|  | Create a Doubly Linked List from a Ternary Tree |  |
|  | Creating a tree with Left-Child Right-Sibling Representation |  |
|  | Convert a given Binary Tree to Doubly Linked List | Set 1 |  |
|  | Convert a given Binary Tree to Doubly Linked List | Set 2 |  |
|  | Convert a given Binary Tree to Doubly Linked List | Set 3 |  |
|  | Convert a given Binary Tree to Doubly Linked List | Set 4 |  |
|  | Convert an arbitrary Binary Tree to a tree that holds Children Sum Property |  |
|  | Convert a Binary Tree to Threaded binary tree | Set 1 (Using Queue) |  |
|  | Convert a Binary Tree to Threaded binary tree | Set 2 (Efficient) |  |
|  | Convert left-right representation of a binary tree to down-right |  |
|  | Convert a given tree to its Sum Tree |  |
|  | Change a Binary Tree so that every node stores sum of all nodes in left subtree |  |
|  | Write an Efficient Function to Convert a Binary Tree into its Mirror Tree |  |
|  | Convert a normal BST to Balanced BST |  |
|  | Convert a Binary Tree into Doubly Linked List in spiral fashion |  |
|  | Convert a Binary Tree to a Circular Doubly Link List |  |
|  | Convert a tree to forest of even nodes |  |
|  | Convert a given Binary tree to a tree that holds Logical AND property |  |
|  | Convert Ternary Expression to a Binary Tree |  |
|  | Lowest Common Ancestor in a Binary Tree | Set 1 |  |
|  | Find distance between two nodes of a Binary Tree |  |
|  | Lowest Common Ancestor in a Binary Search Tree |  |
|  | Print common nodes on path from root (or common ancestors) |  |
|  | Shortest distance between two nodes in BST |  |
|  | Print common nodes on path from root (or common ancestors) |  |
|  | Print the path common to the two paths from the root to the two given nodes |  |
|  | Sqrt (or Square Root) Decomposition |  |
|  | LCA for general or n-ary trees (Sparse Matrix DP approach < O(nlogn), O(logn)>) |  |
|  | Longest Common Extension / LCE |  |
|  | Write a program to Delete a Tree |  |
|  | Write a Program to Find the Maximum Depth or Height of a Tree |  |
|  | Write Code to Determine if Two Trees are Identical |  |
|  | Write a program to Calculate Size of a tree |  |
|  | Root to leaf path sum equal to a given number |  |
|  | How to determine if a binary tree is height-balanced? |  |
|  | Diameter of a Binary Tree |  |
|  | Check for Children Sum Property in a Binary Tree |  |
|  | Program to count leaf nodes in a binary tree |  |
|  | The Great Tree-List Recursion Problem |  |
|  | Given a binary tree, print out all of its root-to-leaf paths one per line |  |
|  | Populate Inorder Successor for all nodes |  |
|  | Connect nodes at same level using constant extra space |  |
|  | Connect nodes at same level |  |
|  | Check if a binary tree is subtree of another binary tree | Set 1 |  |
|  | Check if a given Binary Tree is SumTree |  |
|  | Print Ancestors of a given node in Binary Tree |  |
|  | Get Level of a node in a Binary Tree |  |
|  | Print nodes at k distance from root |  |
|  | Foldable Binary Trees |  |
|  | Maximum width of a binary tree |  |
|  | Double Tree |  |
|  | Given a binary tree, print all root-to-leaf paths |  |
|  | Linked complete binary tree & its creation |  |
|  | Check whether a given Binary Tree is Complete or not | Set 1 (Iterative Solution) |  |
|  | Find the maximum sum leaf to root path in a Binary Tree |  |
|  | Vertical Sum in a given Binary Tree | Set 1 |  |
|  | Sum of all the numbers that are formed from root to leaf paths |  |
|  | Find next right node of a given key |  |
|  | Deepest left leaf node in a binary tree |  |
|  | Extract Leaves of a Binary Tree in a Doubly Linked List |  |
|  | Remove all nodes which don’t lie in any path with sum>= k |  |
|  | Print Left View of a Binary Tree |  |
|  | Check if all leaves are at same level |  |
|  | Find depth of the deepest odd level leaf node |  |
|  | Difference between sums of odd level and even level nodes of a Binary Tree |  |
|  | Custom Tree Problem |  |
|  | Iterative Method to find Height of Binary Tree |  |
|  | Tree Isomorphism Problem |  |
|  | Check if a binary tree is subtree of another binary tree | Set 2 |  |
|  | Find the maximum path sum between two leaves of a binary tree |  |
|  | Threaded Binary Tree |  |
|  | Reverse alternate levels of a perfect binary tree |  |
|  | Print Right View of a Binary Tree |  |
|  | Print all nodes at distance k from a given node |  |
|  | Find distance between two given keys of a Binary Tree |  |
|  | Print all nodes that don’t have sibling |  |
|  | Check if a given Binary Tree is height balanced like a Red-Black Tree |  |
|  | Print all nodes that are at distance k from a leaf node |  |
|  | Find the closest leaf in a Binary Tree |  |
|  | Diagonal Sum of a Binary Tree |  |
|  | Bottom View of a Binary Tree |  |
|  | Print Nodes in Top View of Binary Tree |  |
|  | Serialize and Deserialize an N-ary Tree |  |
|  | Check if a given graph is tree or not |  |
|  | Print nodes between two given level numbers of a binary tree |  |
|  | Find Height of Binary Tree represented by Parent array |  |
|  | Minimum no. of iterations to pass information to all nodes in the tree |  |
|  | Check if two nodes are cousins in a Binary Tree |  |
|  | Find Minimum Depth of a Binary Tree |  |
|  | Maximum Path Sum in a Binary Tree |  |
|  | Expression Tree |  |
|  | Iterative Search for a key ‘x’ in Binary Tree |  |
|  | Find maximum (or minimum) in Binary Tree |  |
|  | Find sum of all left leaves in a given Binary Tree |  |
|  | Remove nodes on root to leaf paths of length < K |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |