**5\_Trees**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level 1** | | | |
| 1. Height of Binary Tree |  | 1. Check for BST |  |
| 1. Determine if two trees are identical |  | 1. Array to BST |  |
| 1. Mirror tree |  | 1. Largest value in each level of binary tree |  |
| 1. Symmetric Tree |  | 1. Maximum GCD of siblings of a binary tr |  |
| 1. Diameter of tree |  | 1. Zigzag Tree Traversal |  |
| 1. Checked for Balanced tree |  | 1. Inorder Successor in BST |  |
| 1. Children Sum Parent |  | 1. Kth Largest Element in a BST |  |
| **Level 2** | | | |
| 1. Check if subtree |  | 1. Maximum sum leaf to root path |  |
| 1. Single Valued Subtree |  | 1. Odd Even Level Difference |  |
| 1. Unique BSTs |  | 1. Lowest Common Ancestor of a Binary Tree |  |
| 1. Inorder Traversal (iterative) |  | 1. Ancestors in Binary Tree |  |
| 1. Preorder Traversal (iterative) |  | 1. Remove BST keys outside the given range |  |
| 1. Postorder Traversal(iterative) |  | 1. Pair with given target in BST |  |
| 1. Vertical Traversal of a Binary Tree |  | 1. Sum Tree |  |
| 1. Boundary Traversal |  | 1. BST to greater sum tree |  |
| 1. Construct Binary Tree from Parent array |  | 1. BST to max heap |  |
| 1. Construct Binary Tree from Preorder and Inorder Traversal |  | 1. Clone binary tree with random pointer |  |
| 1. Preorder Traversal and BST |  | 1. Maximum sum of non adjacent nodes |  |
| 1. Construct tree from preorder traversal |  | 1. Largest BST in a Binary Tree |  |
| 1. Minimum distance between two given nodes |  | 1. Extreme nodes in alternate order |  |
| **Level 3** | | | |
| 1. Connect nodes at same level |  | 1. K-Sum Paths |  |
| 1. Nodes at given distance in a Binary Tree |  | 1. Number of turns in a binary tree |  |
| 1. Sorted Linked List to BST |  | 1. Merge two BST’s |  |
| 1. Binary Tree to Doubly Linked List |  | 1. Fixing two nodes of a BST |  |
| 1. Maximum sum path between two leaf nodes |  | 1. Burn Binary Tree |  |

|  |
| --- |
| Link : <https://www.geeksforgeeks.org/top-50-tree-coding-problems-for-interviews/> |

**5.1 Tree in Data Structure | Introduction to Trees**

* It is a nonlinear data structure
* Its having multiple level
* Trees are used to represent the data items which are having hierarchical relationships between them
* Logical representation of tree in data structure
* By default, the direction of the tree is top to bottom

The notes can contain data

A whiteboard with a diagram and text

Description automatically generatedA whiteboard with text and drawings

Description automatically generatedA white board with writing on it

Description automatically generated

|  |  |
| --- | --- |
| 1. **Root** **:** top most element (the node which does not have any parent ) 2. Elements of trees are known as nodes 3. **Parent Nod**e **:** immediate predecessor 4. **Child Node :** immediate successor 5. **Leaf Node / External Node :** The node which don’t have any child 6. Degrees of the leaf node is 0 7. **Non-leaf Node / Internal Node :** it has at least one child (all the other node except leaf node) 8. **Edge :** Link between two nodes (likes are uni direction) 9. **Path :** It is a sequence of consecutive edges from source node to destination node 10. **Ancestor :** Any predecessor node on the path from root to the node 11. **Descended :** Any successor node on the path from the node to leaf node 12. **Sub-tree:** containing a tree and all of its descendants | 1. **Sibling:** All the children of same parent 2. **Degree:** Number of children of that node 3. **Degree of the tree :** Maximum degree of this tree 4. **Depth of node**: The length of a path from root to that node (number of edges from root to that node)    1. Depth of root is 0 5. **Hight of node:** no of edges in the longest path form that node to a leaf (path between that node to its leaf node) (max distance) 6. **Hight of the tree :** height of root node (Longest path from root node) 7. **Level of node :** number of edges between root to its the given node    1. (each hierarchy known as level) 8. **Level of node =** **Depth of node** 9. **Level of tree = Height of a tree**    1. If the has n nodes the tree will contain (n-1) edges (it can not be a cycle) |

|  |  |
| --- | --- |
| **Binary tree:**  **a**ctual implementation  A white board with writing on it  Description automatically generated | **Application of tree:**   1. File system 2. Routing potocall 3. Organize the data for quick search (for insertion and deletion) |

* 1. = DONE