**BST:**

|  |  |  |
| --- | --- | --- |
|  | Problem | Status |
|  | Construct BST from given preorder traversal | Set 1 |  |
|  | Construct BST from given preorder traversal | Set 2 |  |
|  | Binary Tree to Binary Search Tree Conversion |  |
|  | Convert a BST to a Binary Tree such that sum of all greater keys is added to every key |  |
|  | Sorted Linked List to Balanced BST |  |
|  | Sorted Array to Balanced BST |  |
|  | Transform a BST to greater sum tree |  |
|  | Construct all possible BSTs for keys 1 to N |  |
|  | Convert a BST to a Binary Tree such that sum of all greater keys is added to every key |  |
|  | In-place Convert BST into a Min-Heap |  |
|  | Convert BST to Min Heap |  |
|  | Construct BST from its given level order traversal |  |
|  | A program to check if a binary tree is BST or not |  |
|  | Find k-th smallest element in BST (Order Statistics in BST) |  |
|  | Check if each internal node of a BST has exactly one child |  |
|  | Check for Identical BSTs without building the trees |  |
|  | K’th Largest Element in BST when modification to BST is not allowed |  |
|  | Second largest element in BST |  |
|  | K’th smallest element in BST using O(1) Extra Space |  |
|  | Check if given sorted sub-sequence exists in binary search tree |  |
|  | Check whether BST contains Dead End or not |  |
|  | Check if an array represents Inorder of Binary Search tree or not |  |
|  | Check if two BSTs contain same set of elements |  |
|  | Largest number in BST which is less than or equal to N |  |
|  | C Program for Red Black Tree Insertion |  |
|  | Left Leaning Red Black Tree (Insertion) |  |
|  | Threaded Binary Tree |  |
|  | Threaded Binary Tree | Insertion |  |
|  | Threaded Binary Search Tree | Deletion |  |
|  | Find the node with minimum value in a Binary Search Tree |  |
|  | Total number of possible Binary Search Trees with n keys |  |
|  | Sorted order printing of a given array that represents a BST |  |
|  | Inorder Successor in Binary Search Tree |  |
|  | Print BST keys in the given range |  |
|  | Find the largest BST subtree in a given Binary Tree | Set 1 |  |
|  | Merge Two Balanced Binary Search Trees |  |
|  | Merge two BSTs with limited extra space |  |
|  | Two nodes of a BST are swapped, correct the BST |  |
|  | Floor and Ceil from a BST |  |
|  | Find if there is a triplet in a Balanced BST that adds to zero |  |
|  | Find a pair with given sum in a Balanced BST |  |
|  | Remove BST keys outside the given range |  |
|  | Add all greater values to every node in a given BST |  |
|  | Inorder predecessor and successor for a given key in BST |  |
|  | Given n appointments, find all conflicting appointments |  |
|  | How to handle duplicates in Binary Search Tree? |  |
|  | Data Structure for a single resource reservations |  |
|  | Count BST nodes that lie in a given range |  |
|  | Count BST subtrees that lie in given range |  |
|  | How to implement decrease key or change key in Binary Search Tree? |  |
|  | Print Common Nodes in Two Binary Search Trees |  |
|  | Count inversions in an array | Set 2 (Using Self-Balancing BST) |  |
|  | Replace every element with the least greater element on its right |  |
|  | Find pairs with given sum such that pair elements lie in different BSTs |  |
|  | Find the closest element in Binary Search Tree |  |
|  | Sum of k smallest elements in BST |  |
|  | Maximum element between two nodes of BST |  |
|  | Binary Search Tree insert with Parent Pointer |  |
|  | Largest BST in a Binary Tree | Set 2 |  |
|  | Leaf nodes from Preorder of a Binary Search Tree |  |
|  | Find median of BST in O(n) time and O(1) space |  |
|  | Remove all leaf nodes from the binary search tree |  |
|  | Count pairs from two BSTs whose sum is equal to a given value x |  |
|  | Find distance between two nodes of a Binary Search Tree |  |
|  | Minimum Possible value of |ai + aj – k| for given array and k. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |