Binary Search Tree - GeeksforGeeks

Source: https://www.geeksforgeeks.org/binary-search-tree-data-structure/

Courses Tutorials Practice Jobs DSA Tutorial Interview Questions Quizzes Must Do Advanced DSA System Design Aptitude Puzzles Interview Corner DSA Python Technical Scripter 2026 Explore DSA Fundamentals Logic Building Problems Analysis of Algorithms Data Structures Array Data Structure String in Data Structure Hashing in Data Structure Linked List Data Structure Stack Data Structure Queue Data Structure Tree Data Structure Graph Data Structure Trie Data Structure Algorithms Searching Algorithms Sorting Algorithms Introduction to Recursion Greedy Algorithms Tutorial Graph Algorithms Dynamic Programming or DP Bitwise Algorithms Advanced Segment Tree Binary Indexed Tree or Fenwick Tree Square Root (Sqrt) Decomposition Algorithm Binary Lifting Geometry Interview Preparation Interview Corner GfG160 Practice Problem GeeksforGeeks Practice - Leading Online Coding Platform Problem of The Day - Develop the Habit of Coding DSA Course 90% Refund Binary Search Tree Last Updated : 6 Dec, 2025 A Binary Search Tree (BST) is a type of binary tree data structure in which each node contains a unique key and satisfies a specific ordering property: All nodes in the left subtree of a node contain values strictly less than the node’s value. All nodes in the right subtree of a node contain values strictly greater than the node’s value. This structure enables efficient operations for searching, insertion, and deletion of elements, especially when the tree remains balanced. BSTs are widely used in database indexing, symbol tables, range queries, and are foundational for advanced structures like AVL tree and Red-Black tree . In problem solving, BSTs are used in problems where we need to maintain sorted stream of data. Operations like search, insertion, and deletion work in O(Log n) time for a balanced binary search tree. In the worst-case (unbalanced), these degrade to O(n) . With self-balancing BSTs like AVL and Red Black Trees, we can ensure the worst case as O(Log n). Basics Introduction Applications Insertion , Search and Delete Minimum and Maximum Floor and Ceil Inorder Successor and Inorder Predecessor Handling duplicates in BST Easy Problems Second largest Sum of k smallest BST keys in given Range BST to Balanced BST Check for BST Binary Tree to BST Check if array is Inorder of BST Sorted Array to Balanced BST Check Equal BSTs BST to Min Heap Add all greater values in a BST Check if two BSTs have same elements Medium Problems BST from Preorder Sorted Linked List to Balanced BST Transform a BST to greater sum tree BST to a Tree with sum of all smaller keys Construct BST from Level Order Check for Level Order of BST Max Sum with No Two Adjacent in BST LCA in a BST Distance between Two Nodes k-th Smallest in BST Largest BST in a Binary Tree | Set 2 Remove all leaves from BST 2 sum in BST Max between two nodes of BST Largest BST Subtree 2 Sum in a Balanced BST Fix a BST with 2 swapped Leaf nodes from Preorder of a BST Check if an Array can represent BST Hard Problems All possible BSTs for keys 1 to N In-place Convert BST into a Min-Heap Merge two BSTs K’th Largest in BST without change Check for Subsequence Max Subarray Unique Element Pairs with sum from two BSTs 3 Sum in a Balanced BST Replace with the least greater on right Leaf nodes from Preorder Minimum Possible value of |ai + aj – k| Special two digit numbers in a BST Important Links Quiz on BST Quiz on Balanced BST Practice Problems’ on BST DSA Tutorial GfG 160 : DSA 360° Comment Article Tags: Article Tags: DSA