Data Structure Lab Spot Questions

03-09-2026

1. Given a BST and two integers `L` and `R`, calculate the sum of all nodes whose values lie in `[L, R]`.
2. Given only the preorder traversal of a BST, reconstruct the original BST.
3. Given two BSTs containing `M` and `N` nodes, merge them into a single balanced BST containing all elements.
4. Given the keys 45, 25, 65, 15, 35, 55, 75, 30, 40, 50, 60, construct a BST. Without creating a second tree, mirror it and perform search, insertion, and deletion directly on the mirrored tree.
5. Given an ordinary BST, convert it into an inorder-threaded BST without changing the relative positions of the nodes.

Data Structure Lab Spot Questions — Hints

Range Sum in BST: Use the BST ordering property to prune unnecessary subtrees. If the current node is smaller than L, you need not visit its left subtree; if it is greater than R, you need not visit its right subtree. Add the key only when it lies within [L, R].

Construct BST from Preorder: The first preorder element is the root. Instead of repeatedly inserting elements, think of maintaining valid minimum and maximum bounds for every subtree. Values smaller than the root belong to the left subtree and larger values belong to the right subtree.

Merge Two BSTs into a Balanced BST: Recall that the inorder traversal of a BST is sorted. Obtain the sorted sequences from both trees, merge them as you would merge two sorted arrays, then construct a balanced BST by repeatedly selecting the middle element as the root.

Mirror BST and Operate Directly: Mirror the tree in place by swapping the left and right child pointers at every node. After mirroring, remember that the ordering is reversed: larger keys are on the left and smaller keys are on the right. Therefore, modify the direction used for search, insertion, and deletion accordingly.

Convert BST to Inorder-Threaded BST: During inorder traversal, keep track of the previously visited node. Whenever a left or right child pointer is NULL, it can be used as a thread to the node's inorder predecessor or successor. Use additional flags such as lthread and rthread to distinguish child links from thread links.