Binary Search Tree (BST) Operations: Time & Space Complexity

Operation	Balanced BST	Skewed BST	Iterative vs Recursive	Best Approach & Why?	Time Complexity	Space Complexity
Insert	Recursive (Good)	Iterative (Best)	Iterative (safe), Recursive (readable)	Recursive is cleaner, but Iterative avoids deep recursion	O(log N) / O(N)	O(log N) / O(1)
Search	Recursive (Good)	Iterative (Best)	Iterative (best for large trees)	Iterative is safer since recursion may cause stack overflow	O(log N) / O(N)	O(log N) / O(1)
Delete	Recursive (Best)	Iterative (Not Feasible)	Recursive (better logic handling)	Recursive preferred for handling two-child deletion cases	O(log N) / O(N)	O(log N) / O(1)
Inorder Traversal	Recursive (Good)	Iterative (Good)	Recursive (best for depth < log N)	Recursive is natural, but Iterative needed for large trees	O(N)	O(N) / O(log N)
Preorder Traversal	Recursive (Good)	Iterative (Good)	Recursive (simple), Iterative (for large trees)	Iterative requires explicit stack, recursive is easier	O(N)	O(N)
Postorder Traversal	Recursive (Best)	Iterative (Feasible)	Recursive (better structured)	Iterative is tricky as it requires two stacks	O(N)	O(N)
Level Order (BFS)	Iterative (Best)	Iterative (Best)	Iterative (only feasible approach)	Uses a queue, recursion is not practical	O(N)	O(N)
Height Calculation	Recursive (Best)	Recursive (Not Feasible)	Recursive (natural solution)	Uses simple `1 + max(left, right)`, but deep recursion issue in skewed trees	O(N)	O(log N) / O(N)
Check if BST is Balanced	Recursive (Best)	Recursive (Not Feasible)	Recursive (best approach)	Needs recursion to check height and balance	O(N) / O(N²)	O(log N) / O(N)

When to Use What?

Scenario	Best Approach	Why?
Tree is small or balanced	Use Recursive	Cleaner & more natural
Tree is large or unbalanced	Use Iterative	Prevents deep recursion & stack overflow
Level Order Traversal (BFS)	Use Iterative (Queue-based)	No recursion alternative
Deletion in BST	Use Recursive	Handles all cases efficiently
Performance is a concern	Use Iterative	Uses O(1) space, avoids recursion overhead