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

| Operation | Balanced BST | Skewed BST | Iterative vs Recursive | Best Approach & Why? | Time Complexity | Space Complexity |
|---|---|---|---|---|---|---|
| Insertion | O(log N) | O(N) | Iterative | Iteration avoids stack overflow | O(log N) / O(N) | O(1) Iterative / O(N) Recurs |
| Deletion | O(log N) | O(N) | Iterative | Recursion can cause stack overflow | O(log N) / O(N) | O(1) Iterative / O(N) Recurs |
| Search | O(log N) | O(N) | Iterative | Iteration prevents unnecessary stack usage | O(log N) / O(N) | O(1) Iterative / O(N) Recurs |
| Inorder Traversal | O(N) | O(N) | Recursive | More natural recursive calls | O(N) | O(N) |
| Preorder Traversal | O(N) | O(N) | Recursive | Easier to implement | O(N) | O(N) |
| Postorder Traversal | O(N) | O(N) | Recursive | Easier to implement | O(N) | O(N) |
| Level Order | O(N) | O(N) | Iterative | Uses Queue instead of recursion | O(N) | O(N) |

## When to Use What?

| Scenario | Best Approach | Why? |
|---|---|---|
| Deep BST with high depth | Iterative | Avoids stack overflow |
| Shallow BST with small size | Recursive | Easier and more readable |
| Tree traversal | Recursive | Natural and simple |
| Level order traversal | Iterative | Queue-based approach is more efficient |