📓 Minimum Absolute Difference in BST – Documentation

1. Problem Statement

Given the root of a Binary Search Tree (BST), return the minimum absolute difference between the values of any two different nodes in the tree.

Constraints:

- The number of nodes in the tree is in the range $[2, 10^4]$.
- $0 <= Node.val <= 10^5$
- Tree is a Binary Search Tree (BST).

2. Intuition

The in-order traversal of a BST yields a sorted list of node values. To find the minimum absolute difference, we only need to compare adjacent elements in this sorted sequence, not all possible pairs.

3. Key Observations

- In-order traversal of BST → Sorted values.
- Absolute difference is smallest between two closest numbers in sorted list.
- Comparing all node pairs is inefficient ($O(n^2)$).
- Instead, keep track of the previous node during in-order traversal.

4. Approach

- Use recursive in-order traversal to traverse the BST.
- Track:
    - prev: Previously visited node value.
    - min_diff: Current minimum difference.
- For each node:

    - Compute node.val - prev
    - Update min_diff if this difference is smaller.
- Return min_diff after traversal.

5. Edge Cases

- Only 2 nodes → Direct comparison.
- Very large values → Should not affect result due to absolute difference.
- Unbalanced BST → Still valid due to in-order property.

6. Complexity Analysis

🕒 Time Complexity:

- O(n): Every node is visited once during in-order traversal.

💾 Space Complexity:

- O(h): Stack space due to recursion, where h is the height of the tree.
    - Worst case: O(n) for skewed tree.
    - Best case: O(log n) for balanced tree.

7. Alternative Approaches

- Iterative In-Order Traversal: Use stack to avoid recursion.
- Full Sorted List:
    - Store all values → sort → compare adjacent → O(n log n)
    - Less efficient due to sorting.

8. Test Cases

✅ Test Case 1

- Input: [4,2,6,1,3]
- In-order: [1,2,3,4,6]
- Output: 1

✅ Test Case 2

- Input: [1,0,48,null,null,12,49]
- In-order: [0,1,12,48,49]
- Output: 1

✅ Test Case 3

- Input: [10, 5, 15, null, null, null, 20]
- In-order: [5,10,15,20]
- Output: 5

9. Final Thoughts

- The power of BST + in-order traversal simplifies what could be a brute-force problem.
- Only adjacent values in sorted order matter.
- This problem demonstrates how leveraging structure (BST) leads to optimal solutions.