Minimum Distance Between BST Nodes

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

Example :

```
Input: root = [4,2,6,1,3,null,null]
Output: 1
Explanation:
Note that root is a TreeNode object, not an array.

The given tree [4,2,6,1,3,null,null] is represented by the following diagram:

          4
        /   \
      2      6
     / \
    1   3

while the minimum difference in this tree is 1, it occurs between node 1 and node 2,
also between node 3 and node 2.
```

Note:

1. The size of the BST will be between 2 and `100`.
2. The BST is always valid, each node's value is an integer, and each node's value is different.

Solution 1

Classical inorder traverse. Time complexity O(N). Space complexity O(1)

This question is the same as problem 530.Minimum Absolute Difference in BST. Except that in 530th, we are given a binary search tree with **non-negative values**. However, it seems that it doesn't check any negative case and my solution in cppp get accepted.

C++

```
class Solution {
  public:
    int res = INT_MAX, pre = -1;
    int minDiffInBST(TreeNode* root) {
        if (root->left != NULL) minDiffInBST(root->left);
        if (pre >= 0) res = min(res, root->val - pre);
        pre = root->val;
        if (root->right != NULL) minDiffInBST(root->right);
        return res;
    }
};
```

Java

```
class Solution {
    Integer res = Integer.MAX_VALUE, pre = null;
    public int minDiffInBST(TreeNode root) {
        if (root.left != null) minDiffInBST(root.left);
        if (pre != null) res = Math.min(res, root.val - pre);
        pre = root.val;
        if (root.right != null) minDiffInBST(root.right);
        return res;
    }
}
```

Python

```
class Solution(object):
    pre = -float('inf')
    res = float('inf')

    def minDiffInBST(self, root):
        if root.left:
            self.minDiffInBST(root.left)
        self.res = min(self.res, root.val - self.pre)
        self.pre = root.val
        if root.right:
            self.minDiffInBST(root.right)
        return self.res
```

written by lee215 original link here

Solution 2

```
int min = Integer.MAX_VALUE;
Integer pre = null;
public int minDiffInBST(TreeNode root) {
    check(root);
    return min;
}

private void check(TreeNode node) {
    if (node == null) return;
    check(node.left);
    if (pre != null)
        min = Math.min(min, node.val - pre);
    pre = node.val;
    check(node.right);
}
```

written by tyuan73 original link here

Solution 3

```
def inorder(self, root, in_list):
    if root is None:
        return in_list
    self.inorder(root.left, in_list)
    in_list.append(root.val)
    self.inorder(root.right, in_list)
    return in_list

def minDiffInBST(self, root):
    min_diff = float('inf')
    in_list = self.inorder(root, [])
    for i in range(1, len(in_list)):
        min_diff = min(min_diff, in_list[i] - in_list[i-1])
    return min_diff
```

written by candytgz original link here