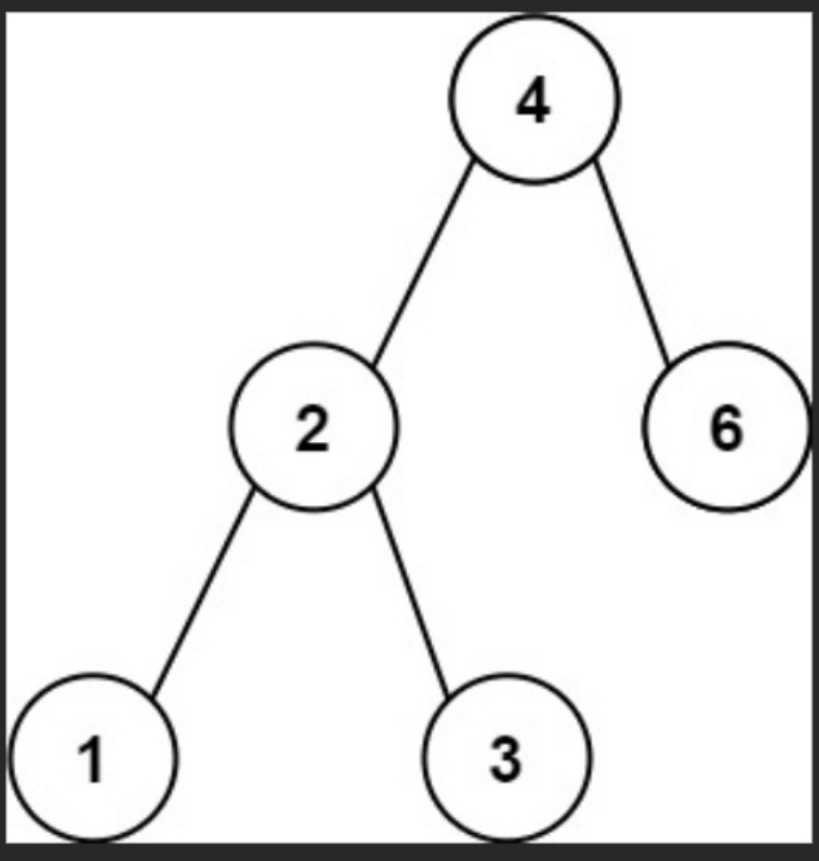
783. Minimum Distance Between BST Nodes

Easy 3.3K 402

Companies

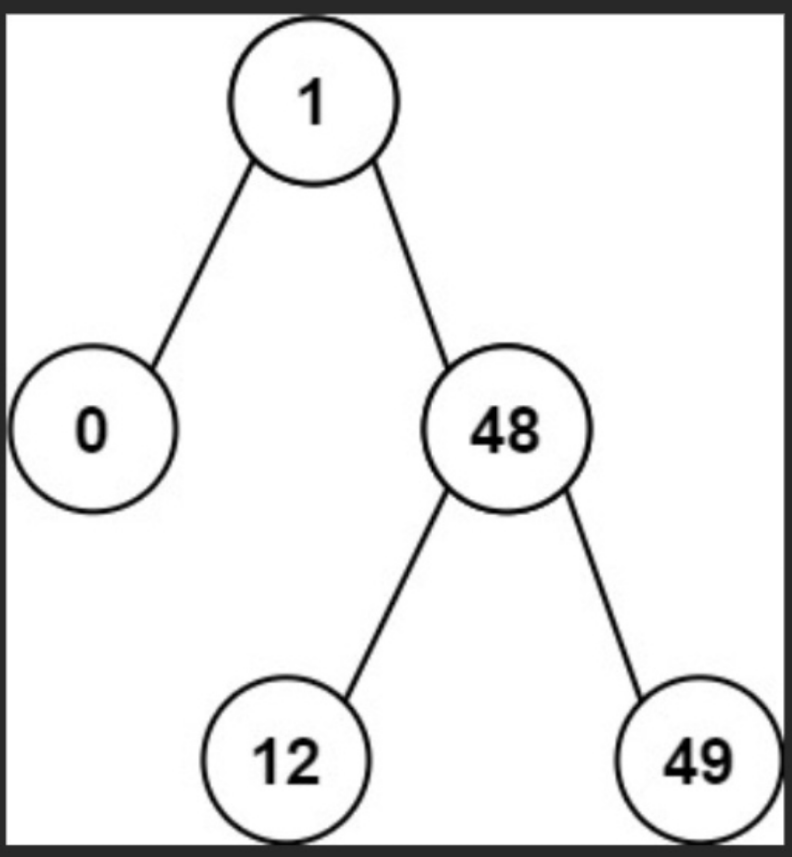
Given the `root` of a Binary Search Tree (BST), return *the minimum difference between the values of any two different nodes in the tree*.

Example 1:




```
Input: root = [4,2,6,1,3]
Output: 1
```

Example 2:



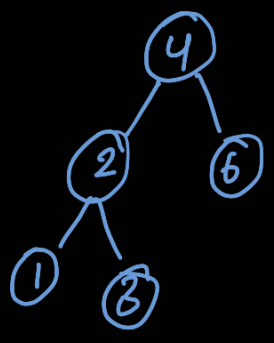
0 1 12 48 49

```
Input: root = [1,0,48,null,null,12,49]
Output: 1
```

Constraints:

- The number of nodes in the tree is in the range `[2, 100]`.
- `0 <= Node.val <= 10^5`

Approach 1:



we know inorder of BST gives increasing order of values. So we find inorder of BST and then check for min distance b/w every two consecutive elements

note: The min difference will not always be between first two nodes of inorder

inorder: 1 4 5 7 9 12

min difference = 1 ie. 5-4

→ Store inorder traversal of BST in a vector
→ Then get min distance by traversing linearly

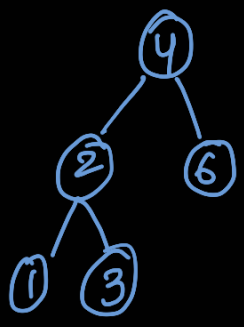
```
for (i : 1 to n-1)
{
    mind = min(mind, a[i] - a[i-1])
}
```

T(n) : O(n) + O(n)
S(n) : O(n)

Approach 2: Without storing the inorder in a seperate vector.

If we see the previous approach, we are only comparing consecutive elements. So when we are at a node

we only need just before visited node in inorder. So we can keep track of that node value while doing inorder and get the minimum difference.




```
prev = -1

find ( root , &mind)
{
    if (root is null)
        return

    find (root->left, mind)
    if (prev == -1)        i.e when the first node in inorder is encountered
        prev = root->val
    else if ((root->val - prev) < mind)
        mind = root->val - prev

    prev = root->val       i.e. Updating prev before moving right.
    find (root->right, mind)
}
```

T(n) : O(n)
S(n) : Recursion stack space