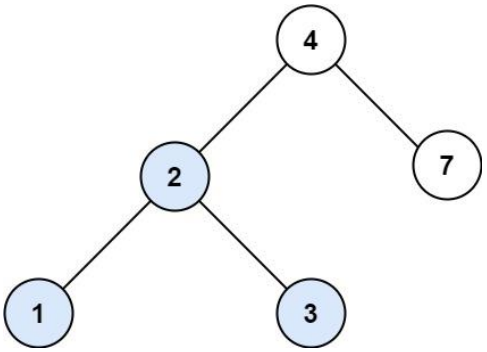
Search in a Binary Search Tree

You are given the `root` of a binary search tree (BST) and an integer `val`.

Find the node in the BST that the node's value equals `val` and return the subtree rooted with that node. If such a node does not exist, return `null`.

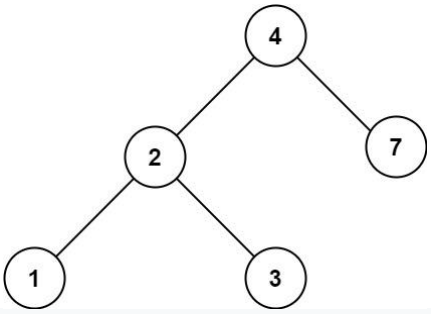
Example 1:




```
Input: root = [4,2,7,1,3], val = 2

Output: [2,1,3]
```

Example 2:




```
Input: root = [4,2,7,1,3], val = 5

Output: []
```

Constraints:

- The number of nodes in the tree is in the range `[1, 5000]`.
- `1 <= Node.val <= 10^7`
- `root` is a binary search tree.
- `1 <= val <= 10^7`

```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     public int val;
 *     public TreeNode left;
 *     public TreeNode right;
 *     public TreeNode(int val=0, TreeNode left=null, TreeNode right=null) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
public class Solution {
    public TreeNode SearchBST(TreeNode root, int val) {
        return Traverse(root,val);
    }

    TreeNode Traverse(TreeNode root, int val)
    {
        if(root == null)
        {
            return null;
        }

        if(root.val == val)
        {
            return root;
        }

        TreeNode l = Traverse(root.left, val);
        if( l == null)
        {
            return Traverse(root.right, val);
        }
        else
        {
            return l;
        }

    }
}
```