776. Split BST

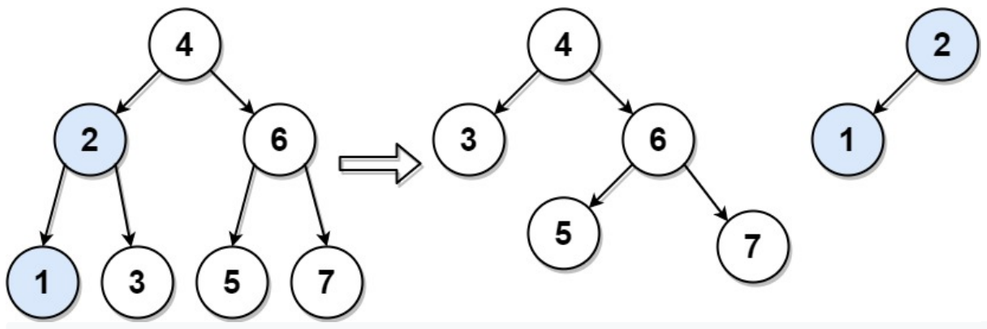
Medium 838 82 Add to List Share

Given the `root` of a binary search tree (BST) and an integer `target`, split the tree into two subtrees where one subtree has nodes that are all smaller or equal to the target value, while the other subtree has all nodes that are greater than the target value. It Is not necessarily the case that the tree contains a node with the value `target`.

Additionally, most of the structure of the original tree should remain. Formally, for any child `c` with parent `p` in the original tree, if they are both in the same subtree after the split, then node `c` should still have the parent `p`.

Return *an array of the two roots of the two subtrees*.

Example 1:




```
Input: root = [4,2,6,1,3,5,7], target = 2
Output: [[2,1],[4,3,6,null,null,5,7]]
```

Example 2:

```
Input: root = [1], target = 1
Output: [[1],[]]
```

Constraints:

- The number of nodes in the tree is in the range `[1, 50]`.
- `0 <= Node.val, target <= 1000`

Accepted 25,089 | Submissions 43,356

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Google | 2

Related Topics

Tree Binary Search Tree Recursion Binary Tree

Similar Questions

Delete Node in a BST Medium

Hide Hint 1

Use recursion. If root.val <= V, you split root.right into two halves, then join it's left half back on root.right.

Java Autocomplete

```java
class Solution {
    public TreeNode[] splitBST(TreeNode root, int V) {
        if (root == null)
            return new TreeNode[]{null, null};
        else if (root.val <= V) {
            TreeNode[] bns = splitBST(root.right, V);
            root.right = bns[0];
            bns[0] = root;
            return bns;
        } else {
            TreeNode[] bns = splitBST(root.left, V);
            root.left = bns[1];
            bns[1] = root;
            return bns;
        }
    }
}
```