Problem 776: Split BST

Problem Information

Difficulty: Medium
Acceptance Rate: 82.32%
Paid Only: Yes
Tags: Tree, Binary Search Tree, Recursion, Binary Tree

Problem Description

Given the `root` of a binary search tree (BST) and an integer `target`, split the tree into two subtrees where the first subtree has nodes that are all smaller or equal to the target value, while the second 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 in order_.

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`

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
vector<TreeNode*> splitBST(TreeNode* root, int target) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public TreeNode[] splitBST(TreeNode root, int target) {
```

```
}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def splitBST(self, root: Optional[TreeNode], target: int) ->
List[Optional[TreeNode]]:
```