Problem 669: Trim a Binary Search Tree

Problem Information

Difficulty: Medium
Acceptance Rate: 66.47%
Paid Only: No
Tags: Tree, Depth-First Search, Binary Search Tree, Binary Tree

Problem Description

Given the `root` of a binary search tree and the lowest and highest boundaries as `low` and `high`, trim the tree so that all its elements lies in `[low, high]`. Trimming the tree should **not** change the relative structure of the elements that will remain in the tree (i.e., any node's descendant should remain a descendant). It can be proven that there is a **unique answer**.

Return _the root of the trimmed binary search tree_. Note that the root may change depending on the given bounds.

Example 1:

![](https://assets.leetcode.com/uploads/2020/09/09/trim1.jpg)

Input: root = [1,0,2], low = 1, high = 2 **Output:** [1,null,2]

Example 2:

![](https://assets.leetcode.com/uploads/2020/09/09/trim2.jpg)

Input: root = [3,0,4,null,2,null,null,1], low = 1, high = 3 **Output:** [3,2,null,1]

Constraints:

* The number of nodes in the tree is in the range `[1, 104]`. * `0 <= Node.val <= 104` * The value of each node in the tree is **unique**. * `root` is guaranteed to be a valid binary search tree. * `0 <= low <= high <= 104`

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:
TreeNode* trimBST(TreeNode* root, int low, int high) {

}
};
```

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 trimBST(TreeNode root, int low, int high) {

}
}
```

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 trimBST(self, root: Optional[TreeNode], low: int, high: int) ->
Optional[TreeNode]:
```