

# Problem 700: Search in a Binary Search Tree

## Problem Information

**Difficulty:** Easy

**Acceptance Rate:** 82.25%

**Paid Only:** No

**Tags:** Tree, Binary Search Tree, Binary Tree

## Problem Description

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 <= 107` \* `root` is a binary search tree. \* `1 <= val <= 107`

## 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* searchBST(TreeNode* root, int val) {

    }
};
```

## 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 searchBST(TreeNode root, int val) {

    }
}
```

### 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 searchBST(self, root: Optional[TreeNode], val: int) ->
        Optional[TreeNode]:
```