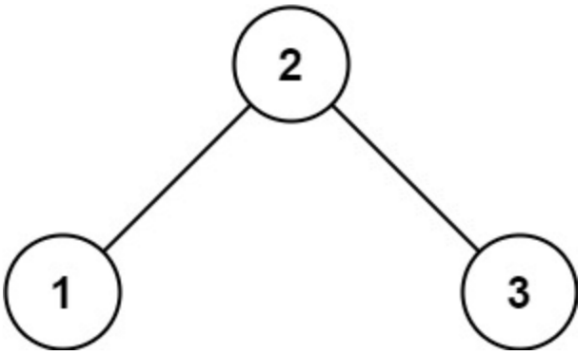
Validate Binary Search Tree

Difficulty	Medium
Category	Tree
Question	https://leetcode.com/problems/validate-binary-search-tree/
Solution	https://www.youtube.com/watch?v=s6ATEkipzow
Status	Done

Question

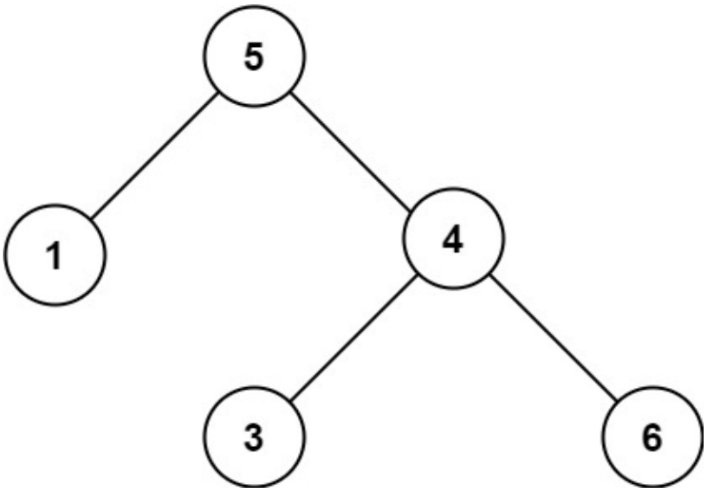
> Given the `root` of a binary tree, *determine if it is a valid binary search tree (BST)*.
>
> A **valid BST** is defined as follows:
>
> - The left of a node contains only nodes with keys **less than** the node's key.
>
> subtree
> - The right subtree of a node contains only nodes with keys **greater than** the node's key.
> - Both the left and right subtrees must also be binary search trees.

Example 1:

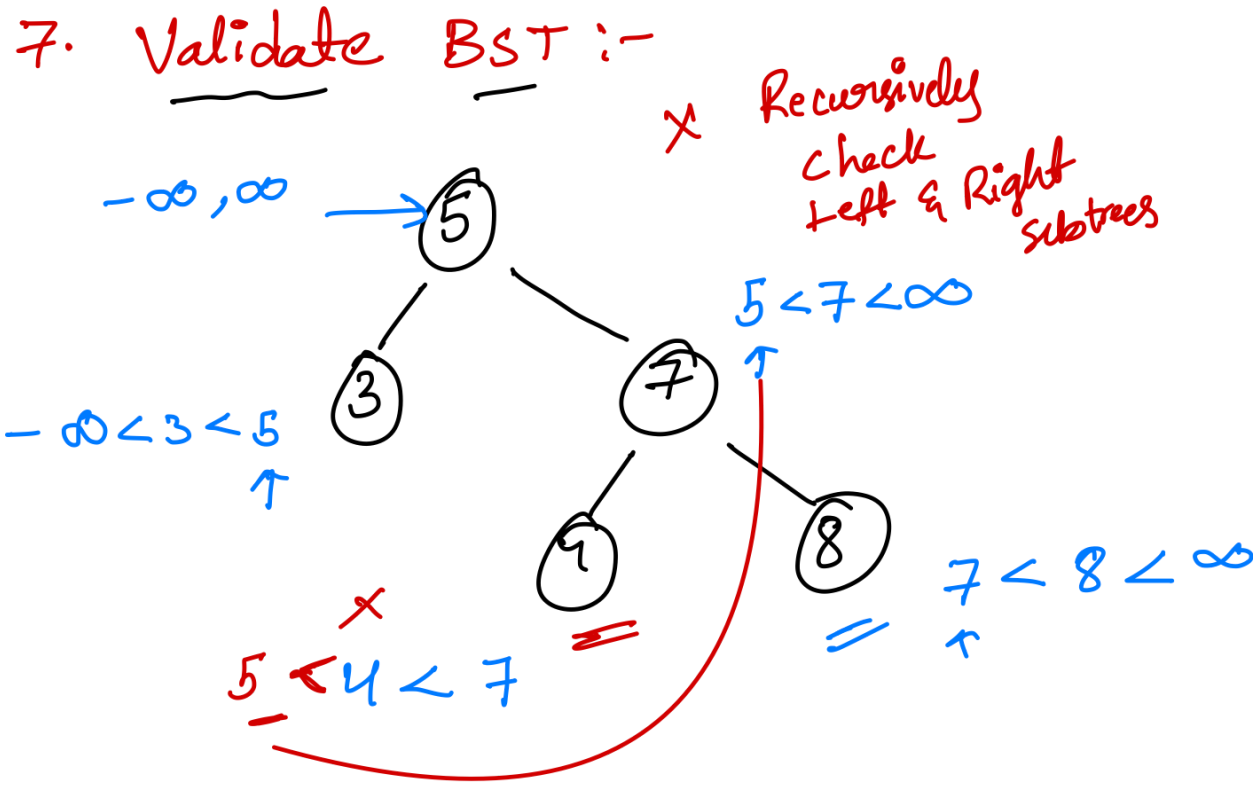


> **Input:** root = [2,1,3]
> **Output:** true

Example 2:



> **Input:** root = [5,1,4,null,null,3,6]
> **Output:** false
> **Explanation:** The root node's value is 5 but its right child's value is 4.



Solution

```
# 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 isValidBST(self, root: Optional[TreeNode]) -> bool:
        # This is a recursive function that checks if a binary search tree is valid.
        def valid(node, left, right):
            # Base case: If the node is None, it's a valid BST.
            if not node:
                return True

            # Check if the current node's value is within the valid range.
            # If not, return False, indicating an invalid BST.
            if not (left < node.val < right):
                return False

            # Recursively check the left and right subtrees.
            # For the left subtree, the valid range is (left, node.val).
```

```
            # For the right subtree, the valid range is (node.val, right).
            return valid(node.left, left, node.val) and valid(node.right, node.val, right)

        # Start the recursion from the root node with an initial valid range of negative infinity to positive infinity.
        return valid(root, float("-inf"), float("inf"))
```

Explanation