JavaScript Validate Binary Search Tree

Challenge

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 subtree of a node contains only nodes with keys less than the node's key.
- 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.

1st Example

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

2nd Example

```
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.
```

Constraints

- `-2³¹ <= Node.val <= 2³¹ - 1`
- The number of nodes in the tree is in the range `[1, 10⁴]`.

Solution

```
const isValidBST = (root, minimum, maximum) => {
    if (root == null) return true;

    if (root.val <= minimum ||
        root.val >= maximum) return false;

    return isValidBST(root.left, minimum, root.val) &&
           isValidBST(root.right, root.val, maximum);
};
```

Explanation

I've defined a function called `isValidBST` that checks whether a binary tree is a valid binary search tree. It takes three arguments: `root`, `minimum`, and `maximum`. The function returns `true` if the tree is a valid binary search tree and `false` otherwise.

The function begins by checking if the `root` is `null`. If it is, it means we have reached the end of the tree, and the function considers it a valid binary search tree. In this case, the function returns `true`.

Next, the function checks if the value of the `root` node is less than or equal to the `minimum` value or greater than or equal to the `maximum` value. If either of these conditions is true, it means the

tree violates the binary search tree property, and the function returns `false`.

If none of the above conditions are met, the function recursively calls itself for the left and right subtrees. For the left subtree, the `minimum` value remains the same, but the `maximum` value is updated to the value of the `root` node. For the right subtree, the `maximum` value remains the same, but the `minimum` value is updated to the value of the `root` node.

Finally, the function returns the logical AND of the results of the recursive calls on the left and right subtrees. This means that both subtrees must be valid binary search trees for the overall tree to be considered valid.

In summary, this function checks if a binary tree is a valid binary search tree by recursively validating the left and right subtrees. It ensures that the values in the left subtree are less than the current node, and the values in the right subtree are greater than the current node. The function returns `true` if the tree satisfies these conditions and `false` otherwise.

Author: Trevor Morin