Q: Check if a given BST is height balanced. Can you do it without processing the BST twice?

A: (See p2.py code)

The algorithm first checks whether a tree exists, since a tree with 0 height is balanced (it satisfies the condition: heights of left/right subtree are <= 1). If a tree does exist, it recursively gets the height of both the left/right subtree by traversing down each side and adding 1 to a counter for every level it encounters for each side (aka the total height).

The difference of both sides is calculated, and the left/right subtrees are checked recursively whether they are balanced (to satisfy the condition that both left and right subtrees of a tree must be balanced).

At the end, the algorithm returns true if the left/right subtree results are balanced, and the difference of heights between the left/right subtrees is <= 1.

As you can see, I process the tree once technically, but realistically I am processing the nodes multiple times each (to check height of total tree, then to check if left sub tree is balanced, I check if the height of left sub tree vs the height of right sub tree is <= 1, etc). My solution is n^2 as I am processing each node in the tree, then for each node I am also processing its subtree.

I think it is possible to solve this in linear time (O(n) and processing the tree once). If you start your processing at the bottom of the tree and return a node's subtree height with respect to its parent node. This will visit all nodes only once and traverse the tree one time. This would be a post order traversal of the tree.