Problem: Implement a binary search tree with an insert method.¶

Constraints

- Can we insert None values?
  - No
- Can we assume we are working with valid integers?
  - Yes
- Can we assume all left descendents <= n < all right descendents?
  - Yes
- Do we have to keep track of the parent nodes?
  - This is optional
- Can we assume this fits in memory?
  - Yes

Test Cases

Insert

Insert will be tested through the following traversal:

In-Order Traversal

- 5, 2, 8, 1, 3 -> 1, 2, 3, 5, 8
- 1, 2, 3, 4, 5 -> 1, 2, 3, 4, 5

If the `root` input is `None`, return a tree with the only element being the new root node.

You do not have to code the in-order traversal, it is part of the unit test.

Algorithm

Insert

- If the root is None, return Node(data)
- If the data is <= the current node's data
  - If the current node's left child is None, set it to Node(data)
  - Else, recursively call insert on the left child
- Else

    - **If the current node's right child is None, set it to Node(data)**
    - **Else, recursively call insert on the right child**

Complexity:

- **Time: O(h), where h is the height of the tree**
    - **In a balanced tree, the height is O(log(n))**
    - **In the worst case we have a linked list structure with O(n)**
- **Space: O(m), where m is the recursion depth, or O(1) if using an iterative approach**