Method Performance

Worst Case Performance:

O(logn) for **insert_element** and **remove_element**:

For every insertion or deletion, the performance is O(logn). The cur_node_height(), get_height(), creation of the node, and self.__balance() are all constant time. The rotations in self.__balance() take a constant number of steps depending on the balance factor. The time complexity comes from the actual insertion and removal itself and the traversal of the tree. Since the tree is balanced after every insertion or removal, the insertion/removal is O(logn) and the traversal of the tree is linear. Therefore, the complexity of the insert_element and remove_element methods are logn + n = O(logn).

O(n) for **to_list**:

The to_list method is linear because it traverses the tree in order, so as n increases, to_list has to recur n times, hence linear. And since each recursion is O(1), to_list runs in linear time O(n)