The worst case performance for insert_element is logarithmic time(O(log n)), which occurs when the size of the data(n) increases exponentially as time increases linearly. The performance is logarithmic because of the method call to  __recursive_insert. The method __recursive_insert traverses the entire tree, from the root down to each leaf node. Because the tree is balanced, the traversal is dependent on the height of the tree, resulting in logarithmic lookup time. The __recursive_insert method also calls the recursive method update_height, which performs at linear time in the worst case because it calls itself until it traverses through the entire tree. The __recursive_insert method also calls __balance, which runs in logarithmic time because it rotates the nodes in the tree until the difference between the right and left subtree of each node is less than or equal to one.

The worst case performance for remove_element is also logarithmic time(O(log n)) due to the method call to __recursive_remove. Because the tree is balanced, the performance is dependent on the height of the tree. The __recursive_remove method calls 3 other methods: locate_min, update_height, and __balance. The method call to locate_min runs in logarithmic time because the method traverses the left side of the tree in order to find the minimum value of the tree. The method update_height also performs in logarithmic time because it must traverse the tree, which varies depending on the height of the tree. Finally, the method call to __balance also performs in logarithmic time because it calls the methods __get_balance, __left_rotation, and __right_rotation. Both rotation methods call update_height, leading to logarithmic performance. The __get_balance method always performs at constant time. After evaluating these method calls, I am confident that remove_element also runs in logarithmic time due to the requirement of traversing through the tree in order to reach the value being removed.

The performance for to_list is at least linear(O(n)) because the method should visit every node in the tree regardless of how balanced the structure is.