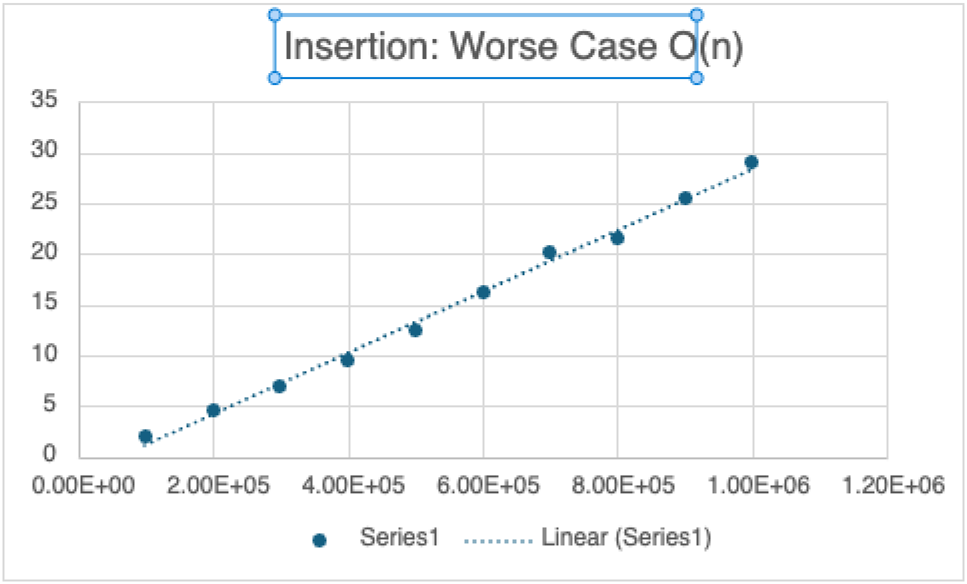
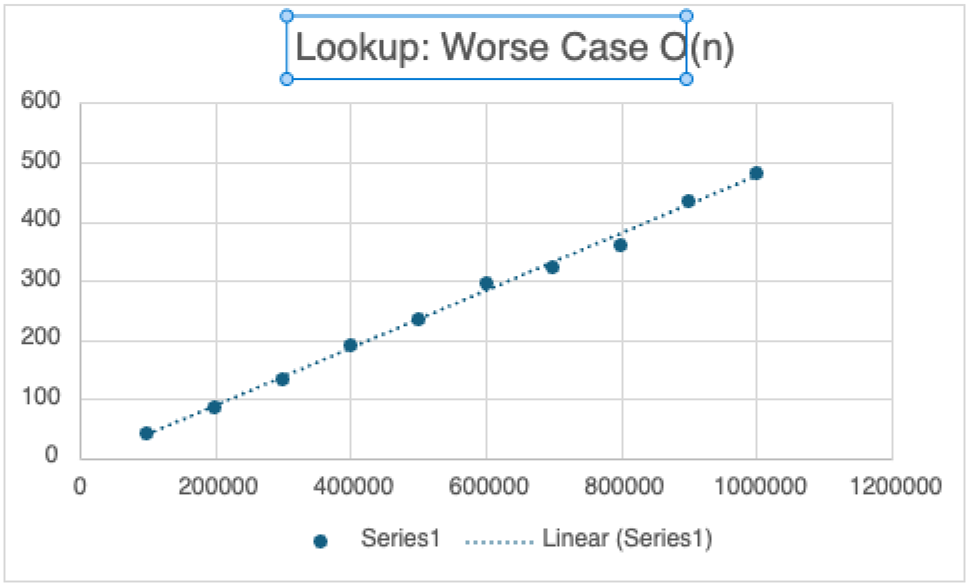


```
def helper(i: BinTree, new_value: Any, comes_before) -> BinTree:
   match i:
       case None:
           return BinarySearchTree(new_value, comes_before, None, None)
       case BinarySearchTree(value, comes_before, left, right):
           if comes_before(new_value, value) is True:
               return BinarySearchTree(value, comes_before, helper(left, new_value,
comes_before), right)
           else:
               return BinarySearchTree(value, comes_before, left, helper(right,
new_value, comes_before))
   return None

def insert(l: BinTree, val: Any, comes_before) -> BinTree:
   return helper(l, val, comes_before)
```

Worst case is linear because it goes through each Node of the Binary Tree to check if it “comes before”, then inserts at the end.



```
def lookup(i: BinTree, look: Any, comes_before) -> bool:
    match i:
        case None:
            return False
        case BinarySearchTree(value, comes_before, left, right):
            if comes_before(look, value) is False and comes_before(value, look) is
False:
                return True
            else:
                if left is None and right is None:
                    return False
                else:
                    return lookup(left, look, comes_before) or lookup(right, look,
comes_before)
    return None
```

Worst case is linear because it checks through each Node of the Binary Tree if the values exist through “comes before” then checks if the value appears in the tree or not.