(1) (25 pts) The add method in IntArrayBag that we discussed in our class.
The worst-case running time is O(n), while the current array are fulfilled with elements, when it need to be added again, it will need to reapply from the memory. Copy original data and putting into new memory, and this will take n steps. So that's why its worst-case running time is O(n).

(2) (25 pts) A method to count the number of occurrences of a particular element target. This method is implemented in the IntArrayBag class that we discussed in class.
The worst-case running time in this method is O(n). The for loop in the method is depend on the length of current linked list. Compile each of the node in the linked list in order, compare the value of current node is equal to the target value or not. So that's why its worst-case running time is O(n).

(3) (25 pts) A method to find a node at a specified position in a linked list starting from the given head. This method is implemented in the IntNode class that we discussed in class.
The worst-case running time in this method is O(n).Compile each of the node in order until "position" node, then return to the current node or the linked list is already reach to the end. This process also depend on introduces to location and compare to the smallest value of the length of the current linked list. The order of compilation has to be from the front to back in order, therefore its running time is O(n).

(4) (25 pts) A method to compute the number of nodes in a linked list starting from the given head. This method is implemented in the IntNode class that we discussed in class.
The worst-case running time in this method is O(n). Compile each of the node in the linked list in order, when it has a node, add the value to 1. when it reach to the end of the linked list means it reaches to the end and return to current count value presenting the length of the linked list. So that's why its running time complexity is O(n).