2.

AVL:

Worst case inserts in O(log n) because in the worst case the tree will be rebalanced [O(1)] and the traversal to the insertion place is O(log n)

Worst case deletes in O(log n) because in the worst case the tree will be rebalanced [O(1)] and the traversal to the node to delete is O(log n)

Worst case counts numbers in range in O(log n)

Maeven and Tzipporah answer:  
Avl tree with an attribute in each node that contains however many guys are

beneath him (the top node contains the size of the tree therefore). Insert goes

down the height of the tree to insert (logn) and adds one to every guy on the

insertion path and does rotations. This runs in the worst case which is log

(n). Delete also works similarly. And countInRange is log(n) if the

implementation is as follows: search for lower bound (logn), on the way

there, count any guys that are bigger than the lower bound and mark them.

Then, add to the count the special attribute of the guys right side (O1)

Similarly, search for upper bound, on the way count the guys who weren’t

counted already who are smaller than the upper bound. (logn). Then finally,

add to the count the special attribute of his left side (O1) Therefore, 2logn

+2O1 = logn.