Jatin Kapoor

Professor Tojeira

CSCI 335 Project 2 Report:

1)

Implementation	Input1.txt	Input2.txt	Input3.txt
Vector	1,925 microseconds	12,208 microseconds	117,138 microseconds
LIst	59,845 microseconds	1,007,432microseconds	32,281,923microseconds
Heap	3,025 microseconds	9,123 microseconds	37,842 microseconds
AVL Tree	2,995 microseconds	12,510 microseconds	59,995 microseconds

Vector: For the vector method, the worst case run-time for this function is O(nlogn). This is because the initial loop has the complexity of O(n), and within the loop you have the time complexity of O(logn). This would lead to the worst case complexity mentioned earlier.
For Popmedian, the worst case is O(n), The average time complexity is O(1).

For vectorMedian, it would be O(n+m), going through instructions, O(N), insert and popMedian would be called within the top. Printing this, it would become O(n+m)

For myAVLtree, the deletetree , which is the helper function, has the complexity of O(N). NN would be the number of nodes in the tree, this performs constant time-operations for each node.

For treeMedian, it is O(mlong(n)). Time complexity of AVL tree insertion and removal is O(longn), n representing the nodes in the AVL tree. The iteration time complexity O(m), and m is the number of instructions. The median would involve finding the max element in the AVL tree findMax. The time complexity would be O(longn). The deletion of the AVl tree has a time complexity of O(n), n representing the number of nodes again. Adding all these and the calculation would lead to the O(mlongn) time complexity as described earlier. In addition for the rotatewithLeftChild, rotatewithRightChild, doublewithLeftChild, and doublewithRightChild, it is O(1). For the remove, findmin, findmax and insert functions it is O(log N).

For myList, the insert is O(n), as the lower bound, long(n), and inserting element, O(n) being combined leads to O(n).

popMedian is O(n), the average is O(1), but worst would be O(n)

For listMedian, the iteration is O(n), the printing medians is O(m), which leads to O(n+m)

myHeap:
The insert is O(log(n)).
For PopMedian, it pops the element from one heap, to rebalance it, which has O(log(n)) for the complexity

For calculateMedian it is O(1)

I have no additional questions.