Q: Observe your output for the BST operations for the build, find (min and max) and delete (min and max) operations.
a. For each operation, compare the worst case complexities of your experimental profiling to the worst case complexity of the theoretical results.

Theoretical complexity for build: $O(n^2)$
This is for when the values being inserted end up being a skew tree.
Worst case complexity for the algorithm I wrote is also $O(n^2)$, as it has the BST add function within a for loop.

Theoretical complexity for findmin, findmax, deletemin, and deletemax: O(n).
This will happen if the tree is completely skewed to the left (for min) or right (for max).
Worst case complexity for the algorithms I wrote are also O(n) for the same reason, but appear O(1) in my results.

b. Justify the worst case complexity of each of your experimental profiling results.
Average case for build is O(nlogn).

Plugging in the values for n and taking ratios, they will be similar to the ratios of the times I got in the build function.

The rest of the functions seem to be O(1), but they are actually O(logn). The thing is, log50000/log10000 < 1.2, so the difference between the times for 10000 and 50000 is going to be very small. Therefore, the differences in times based on input size is more likely to be a result of the random values than the complexity of the find and delete functions.