I set the number of entries to 100,000.

1. Compare the trees in Test A and Test B.

Tree A was built much faster (2971ms vs 64276ms) and had marginally faster processing (0ms vs 6ms). With tree A, left branches only contained duplicates of the current node. With tree B, there were no right branches; this is the slowest possible tree and has growth closer to $O(n^2)$.

2. Compare the trees in Test A and Test C.

Tree C was built in 128ms, which was much faster than Tree A. An important factor is left height vs. right height; Tree C had much closer numbers so it was likely more balanced.

3. Compare the tree in Test C to the shuffled list in Test C.

The tree was faster (0ms vs 284ms). The tree is O(log n) and the list is O(n). With the tree, you'd ideally halve the size of n after each comparison, whereas with a linked list you have to travel the entire list to find all the duplicates.

4. What is the main characteristic of a binary search tree that affects its efficiency?

Balance. You ideally want a BST to be complete so that the max height is as low as possible, since the number of comparisons is equal to the height of the target.