Grant Mercer

Assignment 7

Part B

**Trie** vs **avlTree**:

Beyond a shadow of a doubt a Trie data structure out performs the avlTree in any case for this specific program. The Trie’s ability to contain 26 children dramatically decreases the size of the entire structure and requires much less overhead due to creation and balancing, as the trie is not required to be balanced. The avlTree has a maximum of two children, so reading a dictionary into the avlTree takes a long amount of time and massive amounts of nodes/memory.

One benefit of the avl however is that it’s search remains at O(log n), where as a trie cannot guarantee a O(log n) search time but instead is O(M) based on the keysize. The insert and isPrefix of the avlTree are still O(log n) but do not make up for the tree’s disadvantages.

Space required for the small dict. For each data structure: