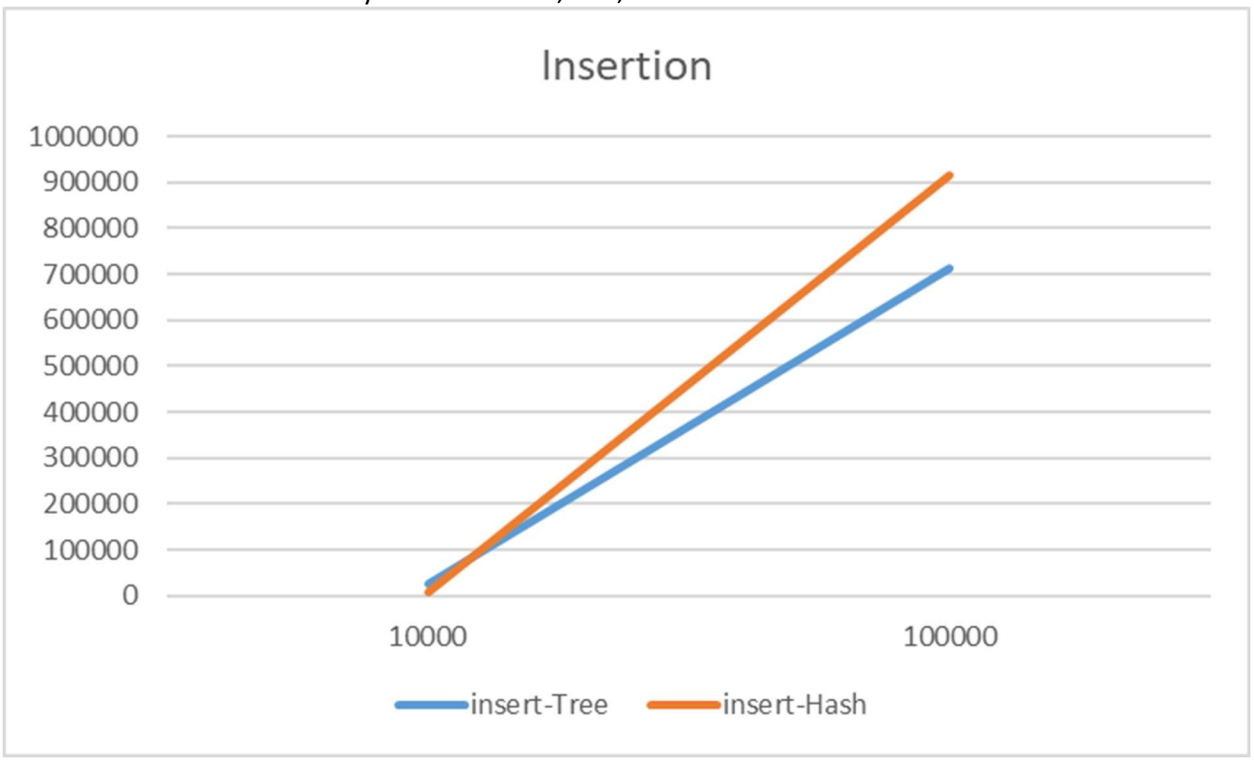
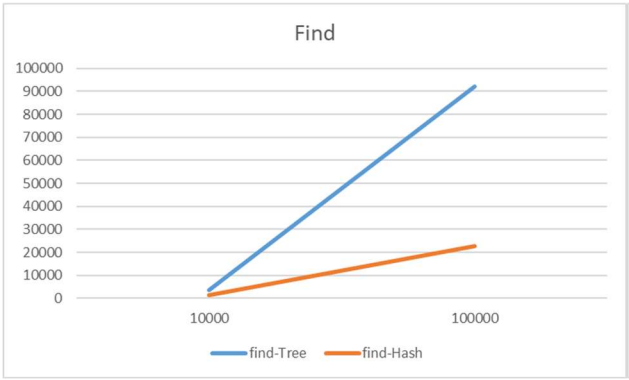
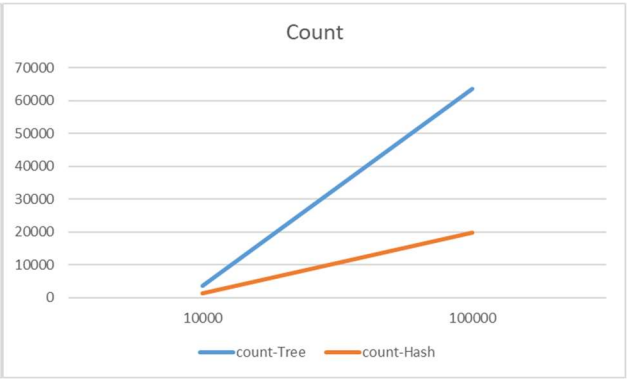
Tree-based vs. Hash-based maps

The three functions that I analyzed were insert, find, and count. The results are below:







The find and count functions scaled as expected, with the constant time hash-based unordered_map scaling far better than the logn time tree-based map. However, the progression of insertion was surprising. Just like the other functions, the hash-table out performs on a dataset of 10,000 items. But unlike the other two functions, the tree-based map outperformed the unordered_map with 100,000 items. This would suggest that the rehashing operations become increasingly computationally expensive as the number of items in the map increases. In practical usage, this would mean that even in situations where ordering is not required, programs with relatively small amounts of data might benefit from using an tree-based map over the hash-based unordered_map if the number of insertions exceed the number of access operations.