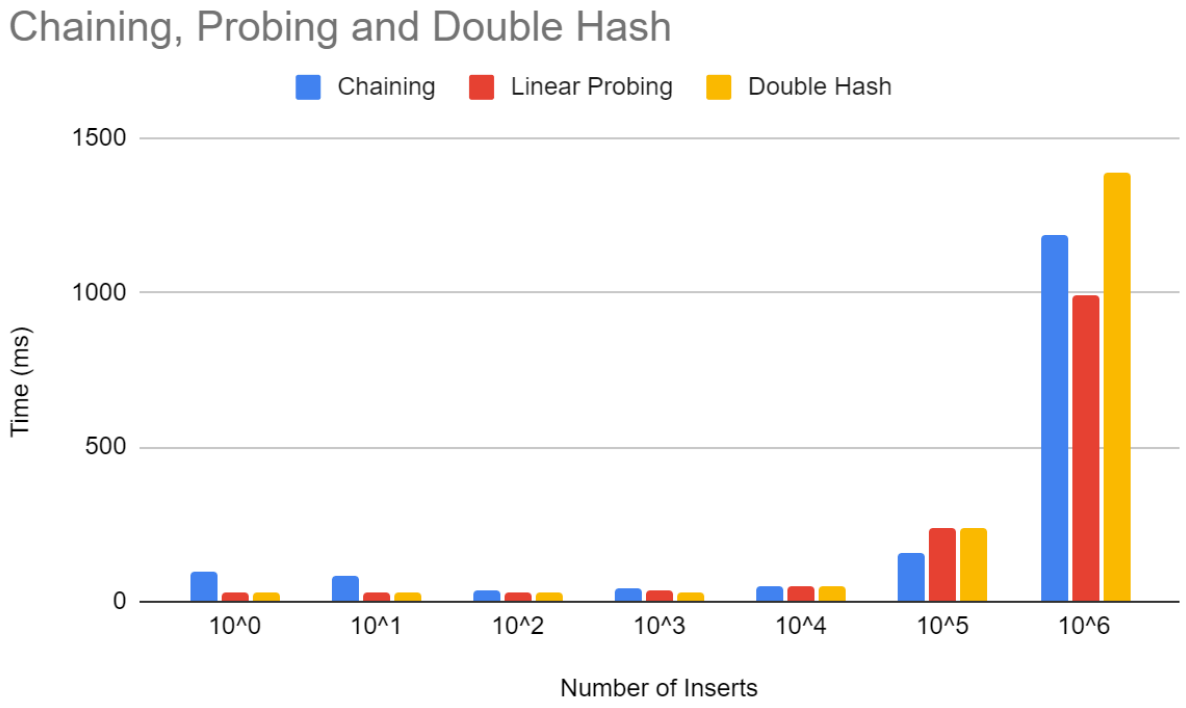
PA4: Carter Cummings

Theoretical Statement:

For each of these hash maps, the time cost will be determined by how collisions are handled. For Chaining it is added to a list at the exact hashing spot, this will result in the desired data always being at a known hash bucket. Getting that hash will take constant time and inserting will take linear time increasing with the number of inserts. Thus chaining is still O(n). Since linear and double hash use the same way to deal with collisions the only difference is how large the step is after a collision. For linear and double once again the hashing is constant. Then a small difference would come with the average time to insert both will be constant time O(1) with double hashing being slightly slower but by a small amount.

Experimental Analysis:



These results did deviate from what was expected for the 10^6 inserts. Double hashing ended up taking about 40% longer than the linear probing which was a small surprise. It was also interesting how time-consuming chaining was at 1 and 10 inserts.

Discussion:

The results were slightly different than expected however it was still seen to be quite efficient. With time divided by the number of inserts ends up going down as the number of inserts increases. The constant time cost of hashing and setting up the arrays ends up decreasing being divided by a larger number of inserts. The overall results showed that linear is always

going to be the best approach except for at 10^5 inserts. Likely with more time invested in optimizing the second hash function that could be improved a lot and would be expected to have less collisions than the linear insert taking less time. However overall this assignment showed how efficient hash tables are at data storage. When implemented right they can have a runtime of O(1) due to jumping right to where the data can be store / found. If it isnt there it can be assumed that the data is close by which will have a minimum number of evaluations before finding where data should be stored / found.