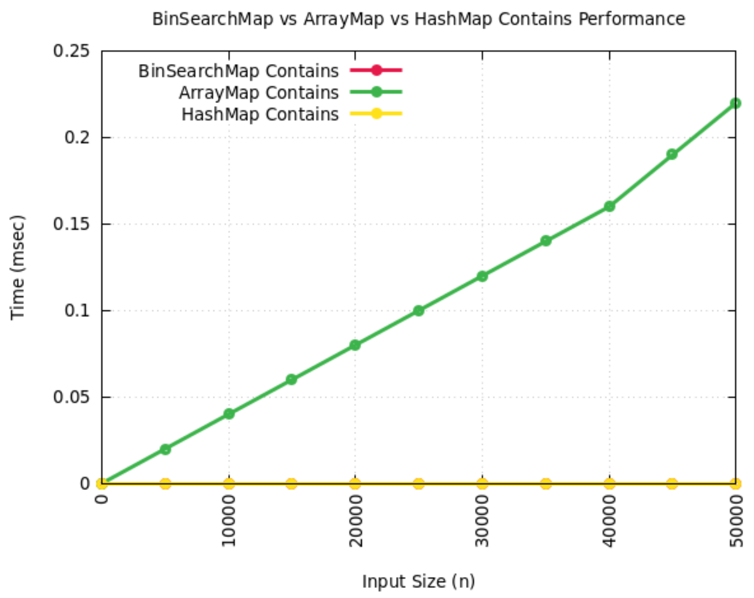
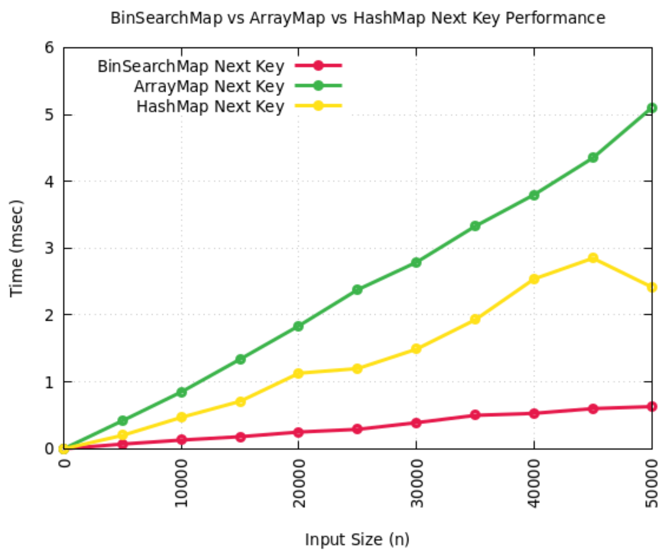
HW7 Writeup (Spring 2022)

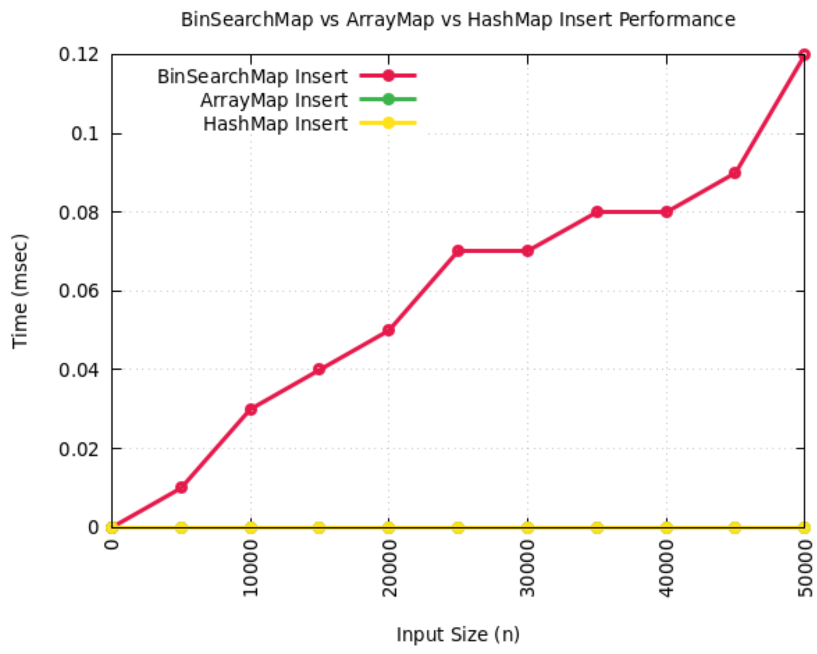
Plots



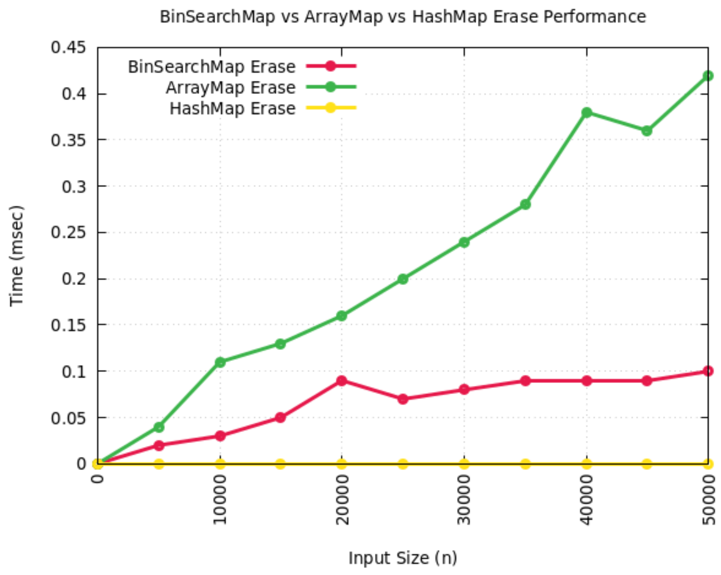
This is likely the most natural graph produced by the performance of my HashMap implementation. The contains function for HashMap appears to be less than O(n) time due to only looping until a key is found and the returning true.



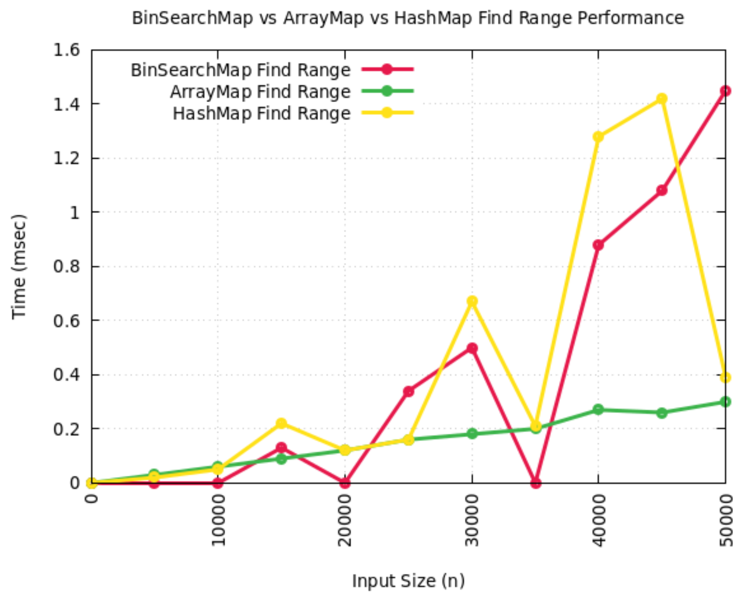
The next_key graph is the next most linear graph produced. It appears to be closer to O(n) as iteration through the entire table is required to find the next consecutive key available among each list in the table of buckets called a Hash Table.



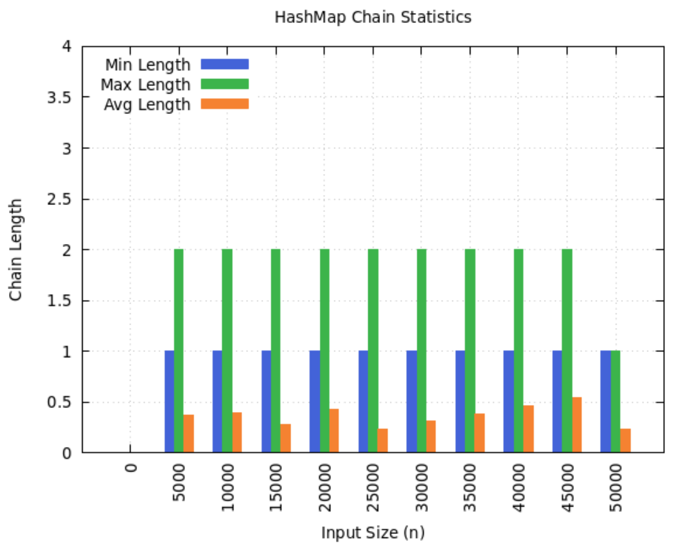
The insert graph also appears to be O(1) for HashMap as it is a constant time function when not needing to be resized. As such, this can be called amortized O(1), averaging out the instances where the table's load factor is reached or exceeded.



The erase graph continues the trend of having HashMap appear as O(1) for removing nodes from the table. As erase has no loops, or occasional calls to resize and rehash, this function truly is performed with a constant time complexity.



This is where things get interesting as the find_keys function in HashMap appears to mirror that of the ArrayMap class. I can assume this has something to do with the unpredictably wide range between two keys that can be chosen. When this range is larger, the function will take more time.



As for the statistics functions, I could not figure out how to implement these such that I passed the tests associated with each chain length function. Therefore, I cannot accurately describe what this graph depicts beyond the distribution of key-value pairs according to my hash function. With that, it appears that the average chain length for chains that are not empty is lower than even the minimum chain length of each input size. Thus, this function is most certainly incorrect.

Challenges:

As previously mentioned, I struggled with completing this assignment. It is 1:00am as I am writing this, and I have been working since around 6pm of the prior day. In the end, I was not passing the final two tests related to chain length (StatsRandom & StatsBiased). The min chain length and max chain length functions seem to have given me the most trouble, especially as was passing all the other tests. In addition, I am quite sure that average chain length was most in error as even my graph of incorrect functions did not make some sort of sense like the plots of other functions in the HashMap class.