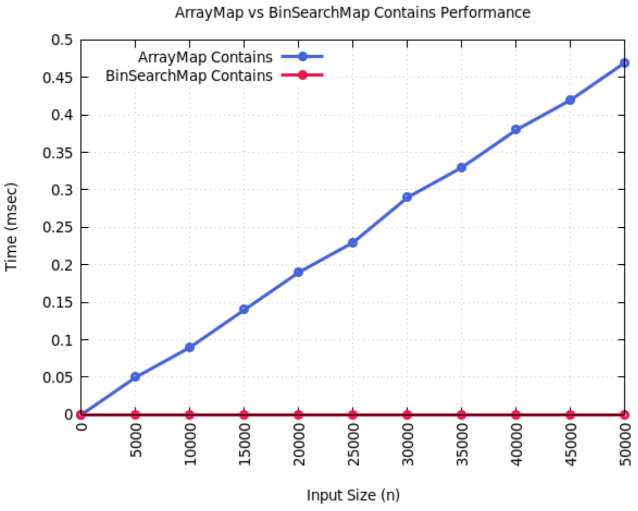
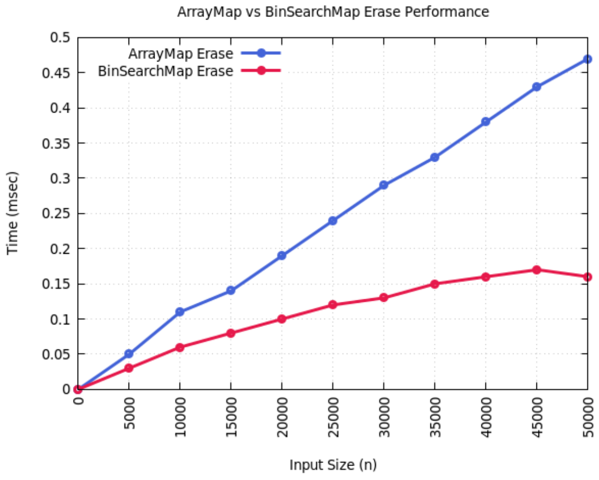
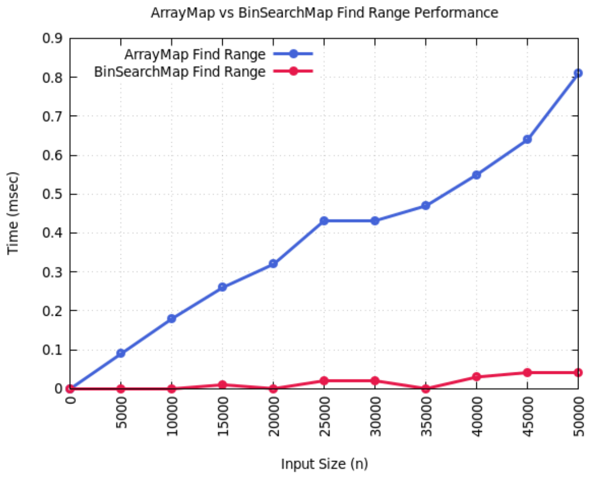
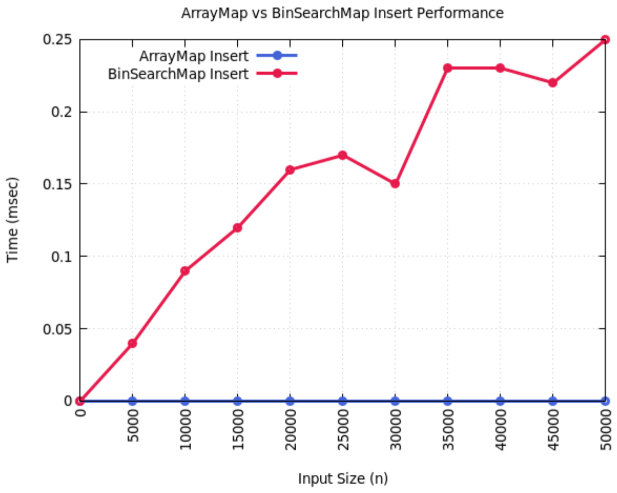
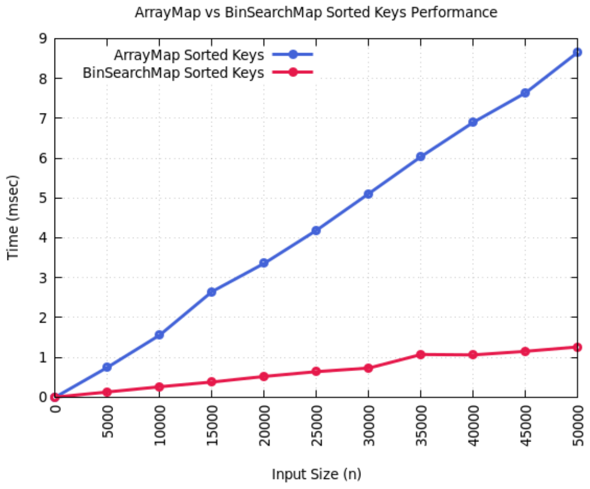
Samuel Sovi
CPSC 223
HW-6

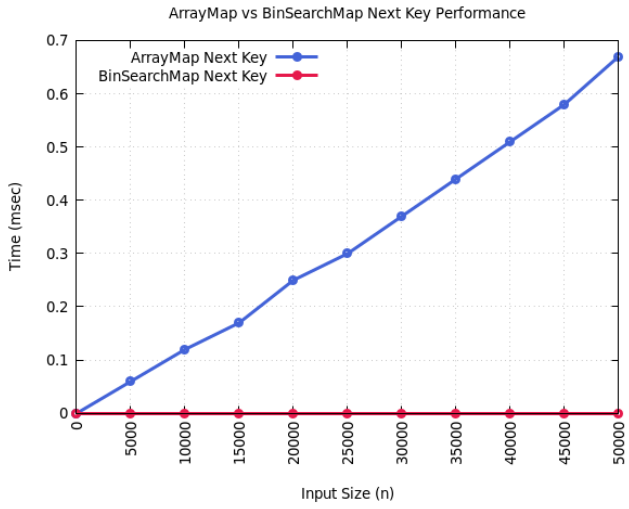












Displayed above are my six graphs that display the performance of my BinarySearchMap and the performance of my ArrayMap. The main idea of the BinarySearchMap is to decrease the time required to perform other operations at the cost of the insert() function. This is due to the nature of a BinarySearchMap which is that it is kept ordered. This means that the insert function must place into the correct order rather than at the end or beginning. As a result, however, the array can get elements quickly since they are in order. In our first graph, we can see that our contains function is much faster in the BinarySearchMap since the BinarySearchMap can perform binary search to find the element while the ArrayMap has to check all keys. Similarly for erase, the BinarySearchMap can find the element much more quickly than ArrayMap but it still must erase the element. Finding the range is also simple because we can quickly get to the start of the desired range using a quick binary sort to find the start rather than go through all the elements to find every element between the given range. Insert is much slower for BinarySearchMaps because we must insert into the correct sorted order. SortedKeys is very fast because the keys are already in sorted order so there is no need to sort again. NextKey is very fast since we can just binary search for the key and find the next index.

In this assignment, I went through quite quickly once my binary search was functional. However, I ran into an issue when I tried to run the performance tests which I soon figured out was because I was not handling the case in which there were no elements in the BinarySearchMap. This was an oversight due to less performance testing, but was remedied due to the number of cases in the performance test. In future assignment I may want to write more test cases of my own in the scenario that I can’t find the error again.