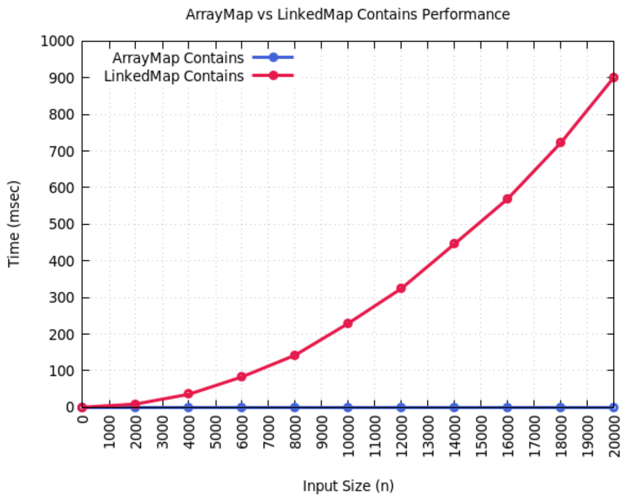
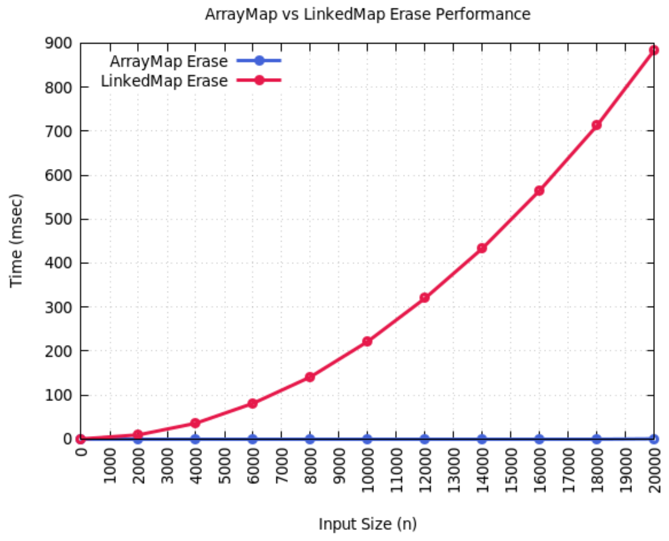
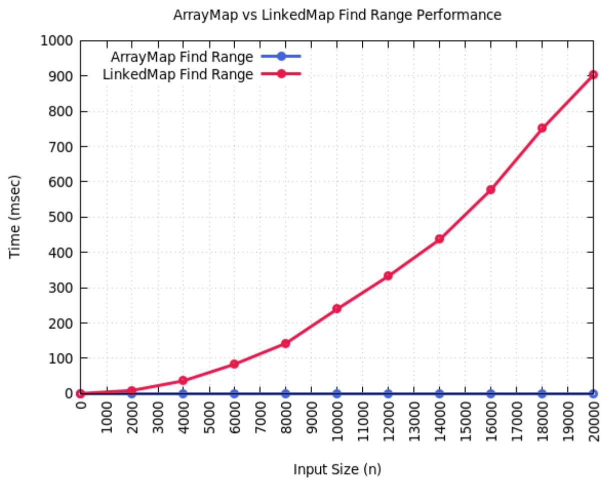
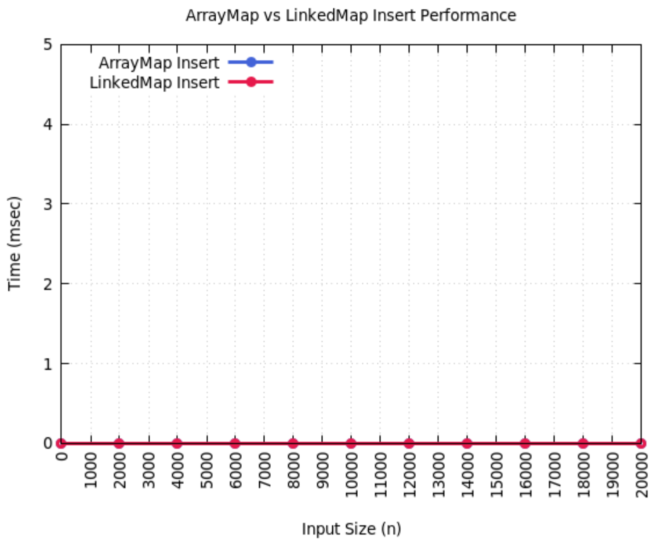
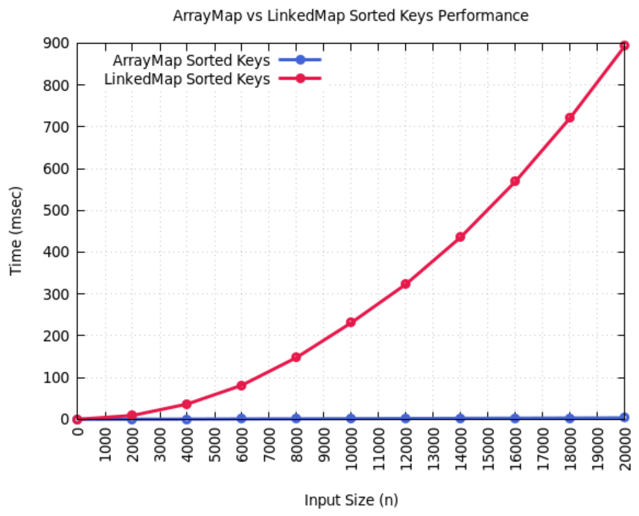
Samuel Sovi
CPSC 223
HW-5

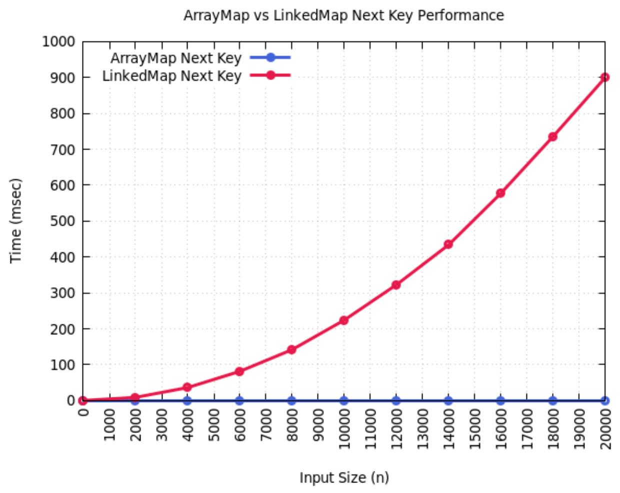












Displayed above are the graphs showing the efficiency of my various Map function implementations using both an ArraySequence and a LinkedSequence. For all the graphs above, we can see that ArrayMap functions are quick since the line is horizontal and seems to have no noticeable slope. However, the LinkedMap's functions all take between 800 and 900 msecs at an input size of 20000 except for the Insert() performance graph. This can be attributed to the fact that array access is O(1) whereas LinkedSeq's access is O(n). However, our implementation for Insert() adds new elements to the end of the Map's sequence of key value pairs. This is O(1) for the Linked Sequence as well because adding to the end of a linked list is O(1) since every node (including the last node) has a pointer to next which can be set to the new element with the tail being updated after.

For the contains method, we just need to check if a key is present which entails looping through an Array or Linked List. The Linked List is significantly slower, however, since each access takes more time as u go further down the list since linked list access is O(n). Our erase method is similar in that it requires iterating through the sequence of key value pairs again and has a similar time for the same reason. For the find_range() function, I need to do multiple comparisons which is slow using a linked sequence, so I used a temporary variable to store the value when appropriate. I used this strategy in my find_next() and find_prev() functions as well. For sorted keys, I once again needed to loop through the sequence and add values to a new ArraySeq of key pairs in sorted order. Lastly, the insert function worked the same in both implementations of my Maps since LinkedSequence has a time of O(1) for inserting at the end.

In this project, I realized that two of my functions took significantly longer than the others in my LinkedMap implementation. These were "find range" and "find next/prev". I later realized that this was because of the fact that I was accessing the same element in my linked sequence multiple times which could be cut by using a temp variable to store the value rather than repeatedly access the linked sequence. Another issue I ran into involved a fault with my ArraySequence Class's move constructor and a lack of deletion of the left hand side. This issue took me a very long time to fix and I lost sleep as a result. However, after finding and fixing the issue, I now have a better understanding of the frustration others feel when I submit code to them that is faulty in the workplace. For cohesion of projects, it is critical that my work works as intended so that others may add their part to it as well. This should prove helpful for me in my coming work experience over the summer.