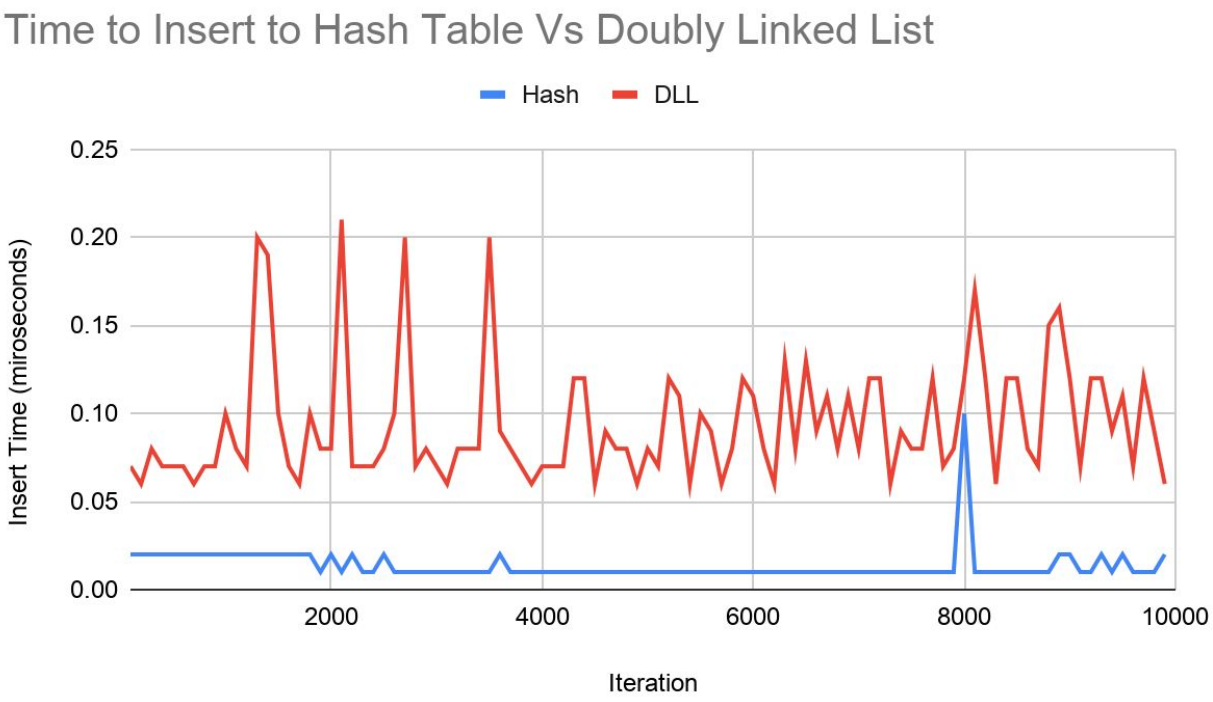
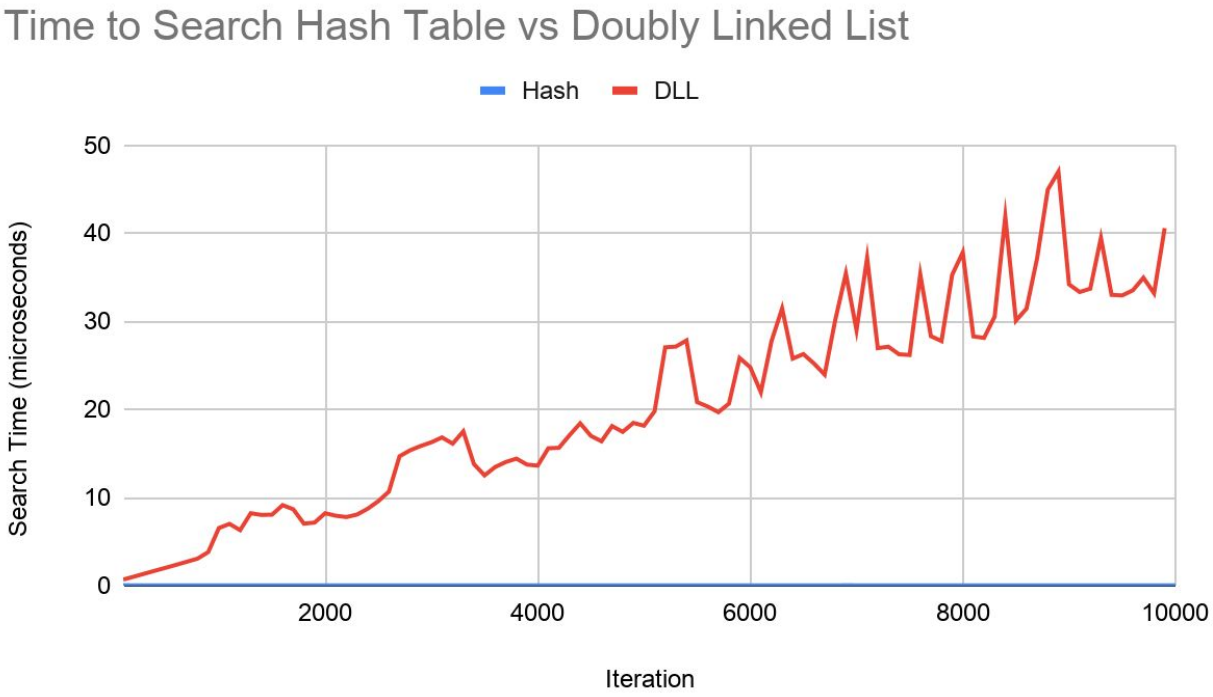
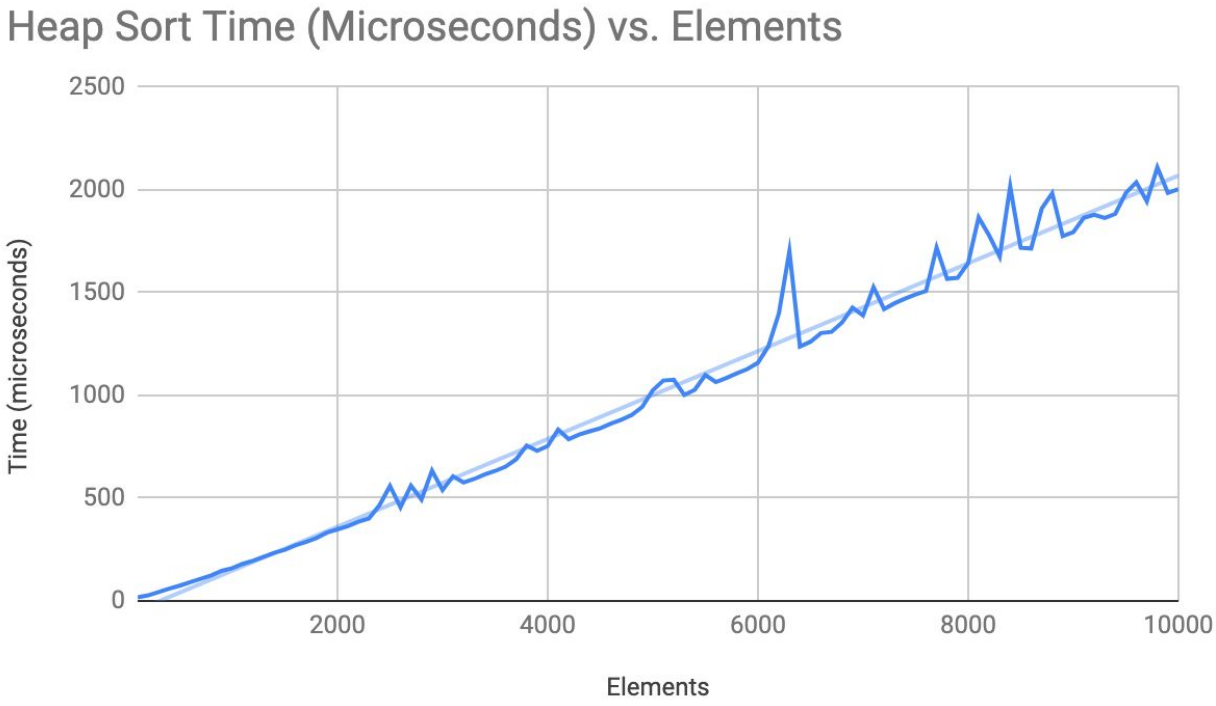
CSCI 2270 Final Project Report

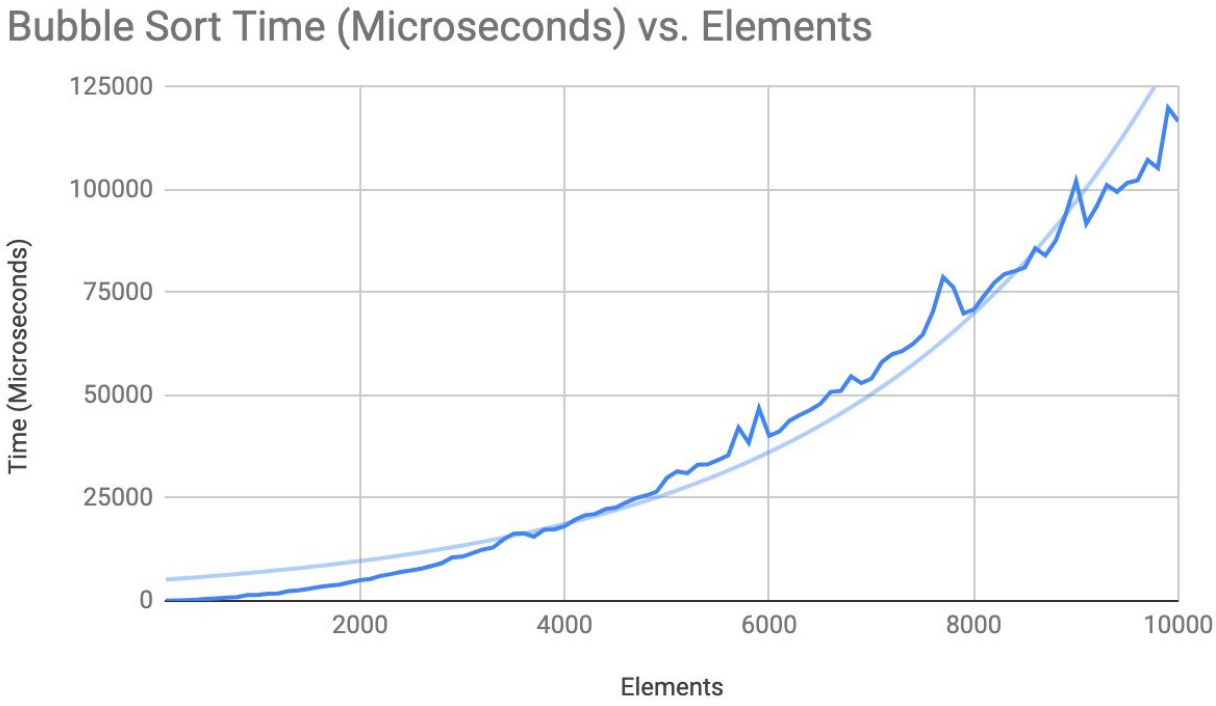
I found that the most efficient data structure for this situation was the hash table by far. The hash table was extremely fast for both operations and the time did not significantly increase as the number of elements increased, showing a O(1) complexity. The linked list on the other hand increased in time, for the search operations, to more than 1000 times longer than the hash table operations and showed a complexity more similar to O(n). The linked list insert times are relatively constant but they are consistently higher than hash table insert times. I believe that this is because of how the hash table is stored in a quickly accessible array while it is much more difficult to access the linked pointers of the linked list.





Between the two sorting algorithms heap sort seemed to perform better for this situation. As shown in the figures below the time taken by the heap sorting algorithm increased in a relatively linear pattern, showing a O(n log(n)) complexity, while the bubble sort algorithm increased in an exponential pattern more similar to a O(n^2) complexity. This shows that the heap sort is much more efficient for larger amounts of data. I believe that this is due to the fact that the larger the dataset that needs to be sorted the more likely it is that the bubble sort algorithm will have to loop through the dataset multiple times and the heap sort algorithm only has to loop through the data twice, once to “heapify” and once to sort, resulting in a much faster sort.


Heap Sort Time (Microseconds) vs. Elements
2500
2000
1500
1000
500
0
Time (microseconds)
2000
4000
6000
8000
10000
Elements


Bubble Sort Time (Microseconds) vs. Elements
125000
100000
75000
50000
25000
0
Time (Microseconds)
2000
4000
6000
8000
10000
Elements