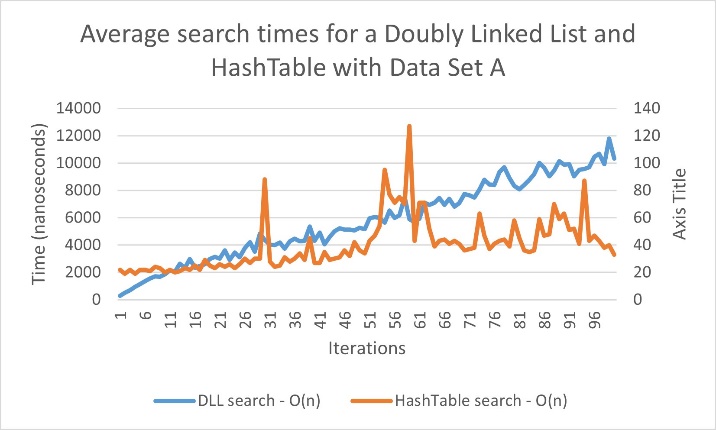
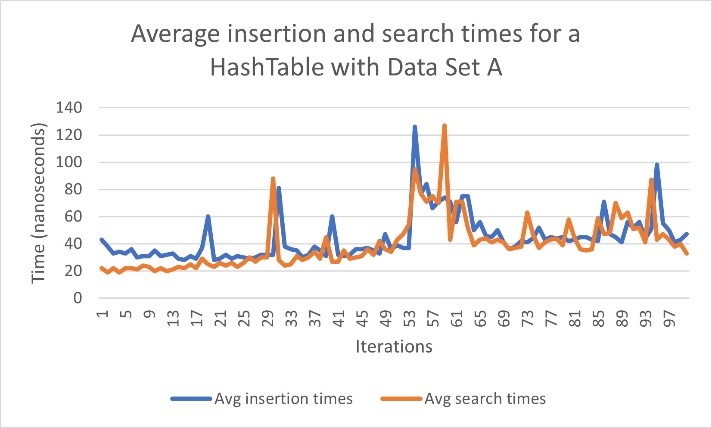
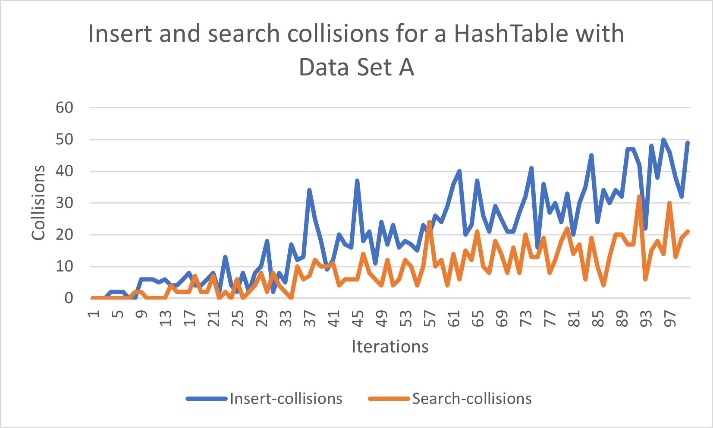
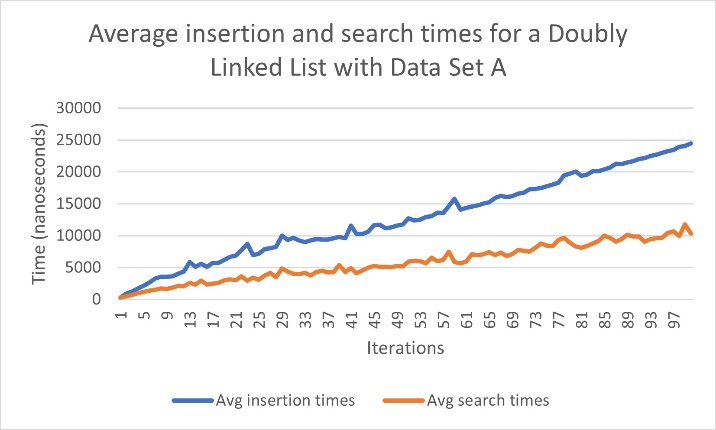
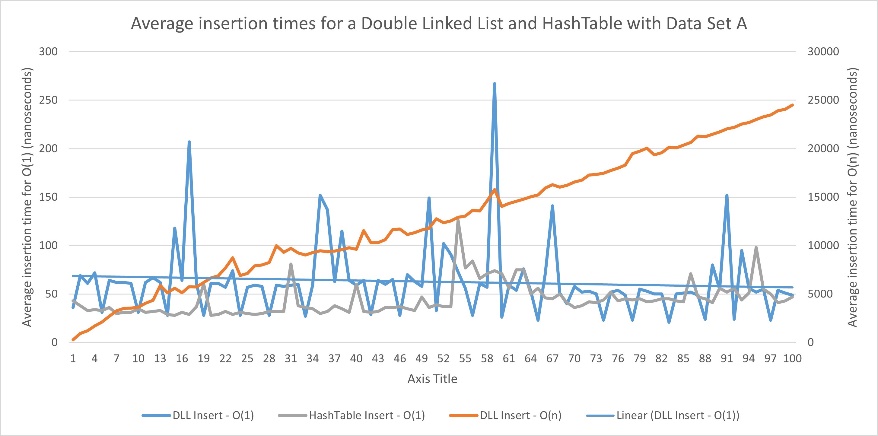
# Part A – Doubly Linked List vs Hash Table

A doubly linked list time complexity when inserting at the head is O(1) because we only need to change some pointers; however, the time complexity for insertion at the tail is O(n) because we must iterate though the entire list to insert a node at the end. The time complexity for search is O(n) because the worst-case scenario is going through the end of the list.

A hash table, offers a time complexity of O(1) for insertion; however as the buckets of the table fills the number of collisions increases and the resulting time complexity for insertion becomes O(n) because the program may have to iterate from 0 to table size to find the last empty bucket. It is possible for the table to have a time complexity of O(1) for search but because there are collisions within the table the time complexity becomes O(n).

A hash table is the best data structure because in a well-designed table we can insert and search for a key with a time complexity of O (1) rather than O(n).





# Part B – Bubble Sort vs Heap Sort

Between the 2 algorithms heap sort is without question the best data structure. A bubble sort algorithm has a time complexity of O(n^2) because we have 2 for loops that iterate together and switches the elements. On the other hand, we have a heap sort algorithm that has a time complexity of O(n \* log(n)) because the time complexity to build a heap is O(n) and to sort from the heap is O(log n). Since O(log n) grows slower than O(n^2), heap sort is the most efficient data structure to use for sorting. In addition, since we are representing our data as a complete binary tree it ensures that the sorting will always be O(log n).

