Simon Chau
SID: 862048772

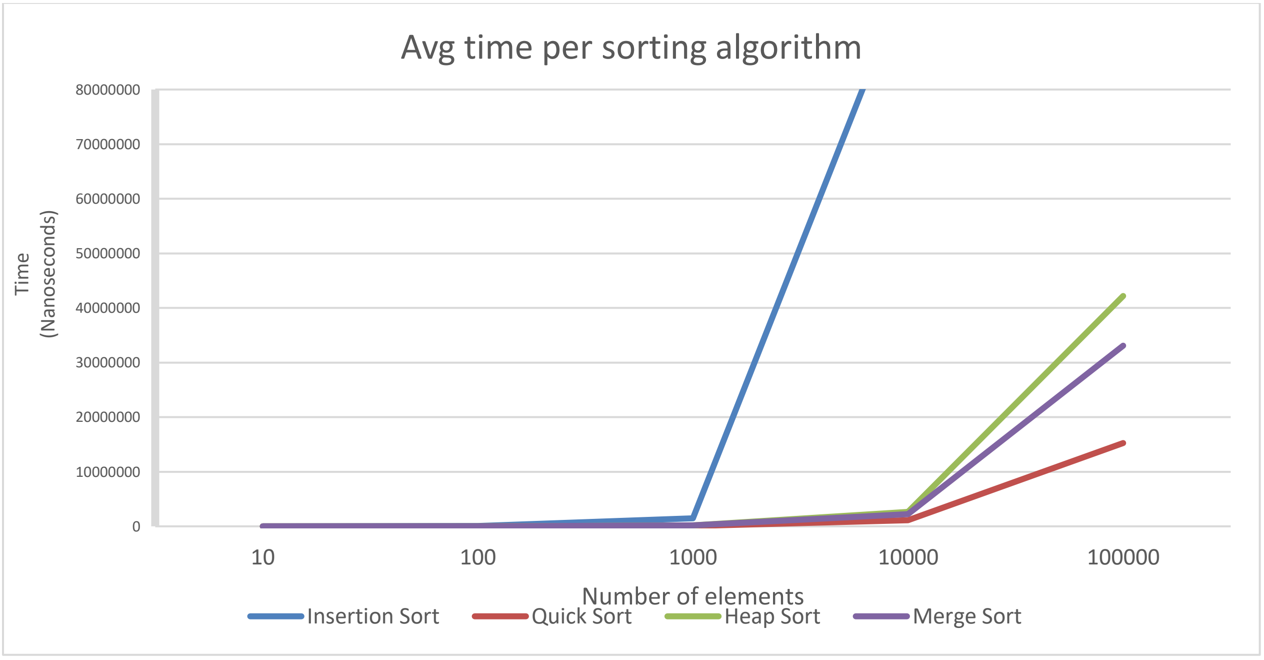
Exercise 1:

If the list is sorted, an array-based implentation of a list is more efficient for binary search since finding a value of a specific node in a linked-list implementation requires traversal by accessing elements sequentially starting from the first node. In an array, binary search is more efficient since it initially checks the middle element, then searches the remaining sublist on the left or right of the middle element. In this case, an array-based implentation works much better since the algorithm can access the middle element of each list/sublist with an asymptomatic complexity of O(1) rather than O(n), which is the complexity of accessing an element through a linked-list.

Exercise 3b:

Average times for sorting algorithms in nanoseconds

Size	Insertion Sort	Quick Sort	Heap Sort	Merge Sort
10	627	867	3034	2953
100	40234	14059	32376	31310
1000	1447820	84214	191807	181545
10000	101046863	1132085	2619441	2256641
100000	12035164025	15254615	42188365	33089590
1000000	N/A	148591609	442751803	312418201



Exercise 3c:

Yes, these times seem reasonable if we look at the graph above. Insertion sort has an average complexity of O(n^2) whereas quick sort, merge sort, and heap sort all have average complexities of O(n log n). The graph above highlights that insertion sort's runtime becomes exponentially larger than the runtimes of the other algorithms as the number of elements increases.