Melina Seng
CTIS 310: Project 7 Results

ArrayList Results:

Number of Elements	Adding each element	Sorting each element	Shuffle each element	Randomly getting each element 1,000,000 times	Sequentially getting each element
1,000	6 ms	3 ms	1 ms	34 ms	0 ms
5,000	8 ms	7 ms	2 ms	44 ms	1 ms
10,000	14 ms	14ms	5 ms	183 ms	2 ms

LinkedList Results:

Number of Elements	Adding each element	Sorting each element	Shuffle each element	Randomly getting each element 1,000,000 times	Sequentially getting each element
1,000	2 ms	1 ms	1 ms	700 ms	1 ms
5,000	5 ms	5 ms	1 ms	11,447 ms	42 ms
10,000	7 ms	14 ms	16 ms	28,794 ms	240 ms

The data collected in the tables show the results for both the ArrayList and LinkedList. While both lists are particularly fast, we can interpret that the ArrayList is typically faster in almost all aspects, besides adding each element. This makes sense because LinkedLists are better for when elements are being added or deleted, since LinkedLists do not have set sizes.

On the other hand, the ArrayList shows that it is much faster than LinkedLists in getting elements, both randomly and sequentially. This can be explained because it is easier to get data from an ArrayList rather than a LinkedList because elements in an ArrayList have an index/memory of where that element is located, whereas elements in a LinkedList do not know where they are in a list, so it is a little more difficult to locate.

Sorting and shuffling the array is almost the same between both lists, however the ArrayList is generally faster.

The results are consistent with how the two lists are implemented. This is because with the ArrayList, it is best when the size does not change much, but we need to access it a lot due to the fixed size with indices for each element. On the other hand, the LinkedList is best when there are lots of data being added or deleted, and we don’t have as many accesses to the data.