For Assignment 3 section 2 Sorting the task was to implement the 3 sort methods Bubble Sort, Merge Sort and Quick Sort and analyze the results using timeit. The results were as follows:

The list that I used was 26 elements in length and consisted of the NATO phonetic alphabet for all 3 sort methods.

alist = [**'Victor'**,**'Zulu'**,**'Charlie'**,**'Mike'**,**'Whiskey'**,**'Quebec'**, **'Golf'**,**'Papa'**,  
 **'Juliet'**, **'Kilo'**,**'Yankee'**,**'Delta'**,**'November'**,  
 **'Oscar'**,**'India'**, **'Foxtrot'**,**'Romeo'**,**'Alpha'**,  
 **'Tango'**,**'Uniform'**,**'Sierra'**, **'Echo'**, **'X-ray'**,  
 **'Lima'**, **'Bravo'**]

1. **When sorted using BubbleSort the result were**:

timeit test, testing bubble\_string\_list.BubbleStringList.sort\_obj()

0  **0.808**7725153894408

1  **0.797**565767368975

2  **0.808**6587610150995

3  **0.812**8565748779848

4  **0.791**8430899905311

1. **When sorted with Merge Sort the results were:**

timeit test, testing bubble\_string\_list.MergeStringList.mergeSort()

0 **0.394**97433174160446

1 **0.402**34533774922265

2 **0.384**57829172559

3 **0.398**173049258141

4 **0.437**45798371203826

1. **When sorted with Quick Sort the results were:**

timeit test, testing bubble\_string\_list.QuickStringList.quickSort()

0 **0.635**031297712677

1 **0.625**9645191782939

2 **0.625**8810068205458

3 **0.636**1974187745214

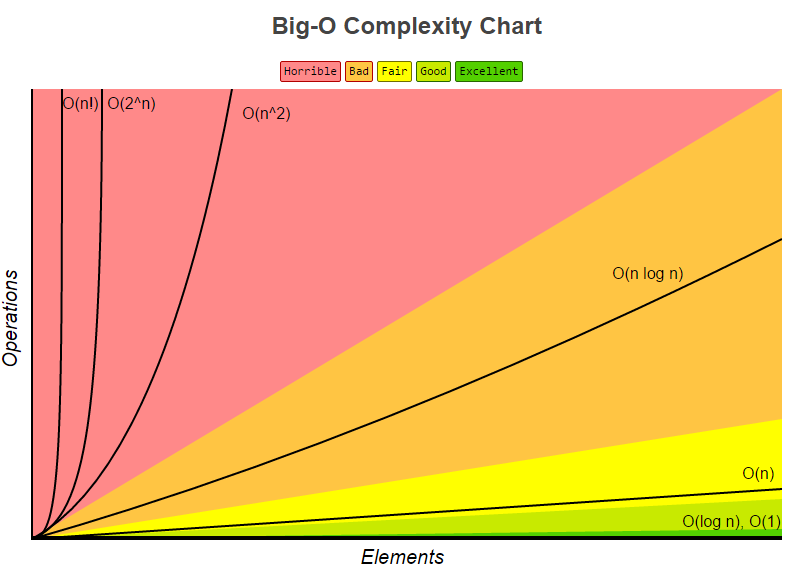
4 **0.627**8367497101343

After testing each sorting method I found that bubble sort was the least efficient of the three methods. After watching the video provided through MyCourses regarding bubble sort and learning that this method is ‘Quadratic’ meaning that Bubble Sorts worst possible scenario has a time complexity of 0(n^2) I referenced back to the ‘Big-O Complexity Chart’ to see whether the results correlated to where it should be, which was true. I then learned that the bubble sort method speed is reduced because it compares each element in the list in pairs and decide which element is smallest as it iterates through the list. For Example, given the provided ‘alist’ list the bubble sort method follows these steps:

1. Compare the first two elements **'Victor'**,**'Zulu'** no switch is necessary because **'Victor'** is smaller than **'Zulu'**
2. Compare second and third elements **'Zulu'** and **'Charlie'** switch because **'Zulu'** larger than **'Charlie'**

This process continues until the end of the list and then repeats until the list is sorted.

The second least efficient method was Quick Sort. I found that for Quick Sort its worst case scenario time complexity is O(n^2) and average is O(n log n). When comparing the time complexity of Quick sort to Merge sort I found that for both worst case scenario and average Merge sort has a time complexity of O(n log n). In my case for Merge sort the method of breaking down the list to its individual element then merging the list back together in order proved to be most efficient.



Quick Sort average case

Quick Sort Worst case

Merge Sort

Bubble Sort