Assignment 2 - Sorting Algorithms

P 1) Comparison on two int to find larger and adding to a new list until we no longer have any remaining in the 2 previous comparison

- List 1: [1, 25, 31, 16] sorted does not require divide and conqrer
- List 2: [-3, 0, 16, 27]

Compare 1 and -3, -3 < 1 [-3] → merge list. 1 stays in its list until a larger int appears. Than compare remaining List 1 and List 2

- List 1 [1, 25, 31, 16]
- List 2 [0, 16, 27]

Compare 1 and 0 0 < 1 [-3|0] (0 is added to merge list)
Repeat process again

- List 1 [1, 25, 31, 16]
- List 2 [16, 27]

Compare 1 and 16, 1 < 16 [-3|0|1] (we now have added 1 from list 1)
Repeat process again

- List 1 [25, 31, 16]
- List 2 [16, 27]

Compare 25 and 16, 25 > 16 [-3|0|1|16] (16 is added from List 2)
Repeat process again

- List 1 [25, 31, 16]
- List 2 [27]

Compare 25 and 27, 25 < 27 [-3|0|1|16|25] (25 is added from List 1)
Repeat process again

- List 1 [31, 16]
- List 2 [27]

Compare 31 and 27, 31 > 27 [-3|0|1|16|25|27] (27 is the last one from List 2)
Repeat process again

- List 1 [31, 16]
- List 2 []

[-3|0|1|16|25|27]

Since we do not have any ints in list 2 we append remaining List 1 to the merge list

Merge List = [-3|0|1|16|25|27|31|16]

P2) Insertion Sort: is the comparison of two ints in a list and to find the larger element and swapping all in the same list

- List [-1, -5, 67, -10, 21, 8, 4, 1] (start with first and second elemen)
  Compare -1 and -5, -5 < -1 swap their place [-5, -1]

- List [-5, -1, 67, -10, 21, 8, 4, 1]
  Move on to the next element (67)
  Compare -1 and 67, -1 < 67. 67 remains in the same place

- List [-5, -1, 67, -10, 21, 8, 4, 1]
  Move on to the next element (-10)
  Compare -10 and 67, -10 < 67. Swap their place
  Compare previors elements -10 < -1 swap, -10 < -5 swap also

- List [-10, -5, -1, 67, 21, 8, 4, 1]
  Move on to the next element (21)
  Compare 67 and 21, 21 < 67. Swap their place
  Compare previors elements -1 < 21 its does not swap

- List [-10, -5, -1, 21, 67, 8, 4, 1]
  Move on to the next element (8)
  Compare 67 and 8, 8 < 67. Swap their place
  Compare previors elements 8 < 21 swap, -1 < 8 does not swap

- List [-10, -5, -1, 8, 21, 67, 4, 1]
  Move on to the next element (4)
  Compare 67 and 4, 4 < 67. Swap their place
  Compare previors elements 4 < 21 swap, 4 < 8 swap, -1 < 4 does Not swap places

- List [-10, -5, -1, 4, 8, 21, 67, 1]
  Move on to the next final element (1)
  Compare 67 and 1, 1 < 67. Swap their place
  Compare previors elements 1 < 21 swap, 1 < 8 swap, 1 < 4 swap
  -1 < 1 Does not swap

After final comparison it should look: [-10, -5, -1, 1, 4, 8, 21, 67]

P3) QuickSort: Choosing a pivot as an element and finding elements greater and lessthan

- List [-5, 42, 6, 19, 11, 25, 26, -3] → Unsorted list
  - Choosing the last element as the pivot [-3]
    Comparing those that are less than [-3] : [-5]
    Comparing those that are greater than [-3] : [42, 6, 19, 11, 25, 26]
    Sort the left and right of pivot [-3] : [-5] [-3] [42, 6, 19, 11, 25, 26]

- Selecting the right side and last element as a pivot [26]
  Comparing those that are less than [26] : [6, 19, 11, 25]
  Comparing those that are greater than [26] : [42]
  Sort the left and right of pivot [26] : [6, 19, 11, 25] [26] [42]

- Selecting the left side and the last element as a pivot [25]
  Comparing those that are less than [25] : [6, 19, 11]
  Comparing those that are greater than [25] : None are greater
  Sort the left and right partitions of pivot [25] : [6, 19, 11] [25] [ ]

- Selecting the left side and the last element as a pivot [11]
  Comparing those that are less than [11] : [6]
  Comparing those that are greater than [11] : 19
  Sort the left and right partitions of pivot 11 : [6] [11] [19]

There is no more elements left to compare, so all there is to do is combine all partitions

[-5, -3, 6, 11, 19, 25, 26, 42]

p47) Shell Sort: Choosing a gap sequence at a certain position and swaping them if the element on the left is greater than the right element

[15, 14, -6, 10, 1, 15, -6, 0] → Unsorted

Comparing elements at 0 and 4: 15 and 1 Swap
Comparing elements at 1 and 5: 14 and 15 No swap
Comparing elements at 2 and 6: -6 and -6 No swap
Comparing elements at 3 and 7: 10 and 0 swap
List [1, 14, -6, 0, 15, 15, -6, 10]

Comparing elements at 0 and 2: 1 and -6 swap
Comparing elements at 1 and 3: 14 and 0 swap
Comparing elements at 2 and 4: -6 and 15 No swap
Comparing elements at 3 and 5: 0 and 15 No swap
Comparing elements at 4 and 6: 15 and -6 Swap
Comparing elements at 5 and 7: 15 and 10 Swap
List [-6, 0, 1, 14, -6, 10, 15, 15]

Using insertion sort on the list

- List [-6, 0] } Compared and they did not swap
- List [-6, 0, 1] }
List [-6, 0, 1, 14] }

- List [-6, 0, 1, 14, -6, 10, 15, 15]

-6 < 14 swap, -6 < 1 swap, -6 < 0 swap -6 = -6 doesn't matter pos.

- List [-6, -6, 0, 1, 14, 10, 15, 15]

10 < 14 swap 14 < 10 does not swap

- List [-6, -6, 0, 1, 10, 14, 15, 15]

14 < 15 does not swap and other 15 also doesn't swap

List: [-6, -6, 0, 1, 10, 14, 15, 15]

P5) Ranking

1 QuickSort = $O(n \log n)$ is one of the fastest for the $O(\log n)$ but worse case is $O(n^2)$

2 MergeSort $O(n \log n)$ in both performance and used for stability

3 HeapSort = $O(n \log n)$ Similar to mergeSort but less stable

4 ShellSort = $O(n \log^2 n)$ One of the most slowest but still better than InsertionSort and BubbleSort

5 InsertionSort = $O(n^2)$ is the average and worse case. It uses fewer swaps

6 BubbleSort = $O(n^2)$ Similar to insertionSort in both average and worse case

P10)

In my studies I found that the sorting algorithms relative success matched ther expected spatial complexity, with a few exceptions brought on by practical issues. In comparison to more effective algorithms like Merge Sort, Quick sort, and Shell Sort, algorithm of $O(n^2)$ difficulty like bubbleSort Selection Sort, and Insertion Sort performeed more worse on bigger inputs. While asymptotic analysis offers an ideal order, system architecture, input characters, and implementation. Quick sort, for example exhibited some variability based on pivot choices but typically functioned well.