Problems:

1 (text) [2 points] Show the step-by-step process of merging two sorted lists in using the mergesort algorithm. The sorted lists are: [1, 25, 31, 16] and [-3, 0, 16, 27]

[1, 16, 25, 31] and [-3, 0, 16, 27]

Step 1: [1, 16, 25, 31] [-3, 0, 16, 27]

1 > -3 → New Array = [-3]

Step 2: [1, 16, 25, 31] [-3, 0, 16, 27]

1 > 0 and 0 > -3 → [-3, 0]

Step 3: [1, 16, 25, 31] [-3, 0, 16, 27]

16 > 1 and 1 > 0 → [-3, 0, 1]

Step 4: [1, 16, 25, 31] [-3, 0, 16, 27]

16 = 16 and 16 > 1 [-3, 0, 1, 16, 16]

Step 5: [1, 16, 25, 31] [-3, 0, 16, 27]

27 > 16 and 27 > 16 → [-3, 0, 1, 16, 27]

Step 6: [1, 16, 25, 31] [-3, 0, 16, 27]

31 > nothing because we cannot find another element larger than 31.

→ [-3, 0, 16, 16, 27, 31]

2 (text) [2 points] Show the step-by-step process of sorting a list using the insertion sort algorithm. The unsorted list is: [-1, -5, 67, -10, 21, 8, 4, 1]

Unsorted list: [-1, -5, 67, -10, 21, 8, 4, 1] Portion to be sorted. result of sorting

Step 1: [-1, -5, 67, -10, 21, 8, 4, 1]

-1 is sorted. [-1, -5, 67, -10, 21, 8, 4, 1]

Step 2: [-1, -5, 67, -10, 21, 8, 4, 1]

-1 > -5 insert -1 → [-5, -1, 67, -10, 21, 8, 4, 1]

Step 3: [-5, -1, 67, -10, 21, 8, 4, 1]

[-5, -1, 67] is sorted [-5, -1, 67, -10, 21, 8, 4, 1]

Step 4: [-5, -1, 67, -10, 21, 8, 4, 1]

67 > -10, -1 > -10, -5 > -10, insert → [-10, -5, -1, 67, 21, 8, 4, 1]

Step 5: [-10, -5, -1, 67, 21, 8, 4, 1]

21 > -10, 21 > -5, 21 > -1, 21 < 67 swap → [-10, -5, -1, 21, 67, 8, 4, 1]

Step 6: [-10, -5, -1, 21, 67, 8, 4, 1]

8 > -10, 8 > -5, 8 > -1, 8 < 21 insert → [-10, -5, -1, 8, 21, 67, 4, 1]

Step 7: [-10, -5, -1, 8, 21, 67, 4, 1]

4 > -10, 4 > -5, 4 > -1, 4 < 8 insert → [-10, -5, -1, 4, 8, 21, 67, 1]

Step 8: [-10, -5, -1, 4, 8, 21, 67, 1]

1 > -10, 1 > -5, 1 > -1, 1 < 4 insert → [-10, -5, -1, 1, 4, 8, 21, 67]

Step 9: [-10, -5, -1, 1, 4, 8, 21, 67]

Array sorted.

3 (text) [2 points] Show the step-by-step process of sorting a list using the quicksort sort algorithm. The unsorted list is: [-5, 42, 6, 19, 11, 25, 26, -3]

Quick sort: [-5, 42, 6, 19, 11, 25, 26, -3] using median as pivot.

Beg = -5 Last = -3 mid = 19 Pivot = -3 ▨ = Sorted parted

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

[-5, -3] ≤ -3 < [42, 6, 19, 11, 25, 26] Step 1

Beg = -5 Last = -3
mid = -5 pivot = -5

Beg = 42 Last = 26 Mid = 19
Pivot = 26

[-5] ≤ -5 < [-3] [6, 19, 11, 25, 26] ≤ 26 < [42] Step 2

beg = 6 Last = 26 Mid = 11 Pivot = 11

[6, 11] ≤ 11 < [19, 25, 26] Step 3

Beg = 6 Last = 11 mid = 6
Pivot = 6

Beg = 19 Last = 26 mid = 25
Pivot = 25

[6] ≤ 6 < [11] [19, 25] ≤ 25 < [26] Step 4

Beg = 19 Last = 25 mid = 19
Pivot = 19

[19] ≤ 19 < [25] Step 5

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

4 (text) [2 points] Show the step-by-step process of sorting a list using the shell sort algorithm. The unsorted list is: [15, 14, -6, 10, 1, 15, -6, 0]

Shell Sort: [15, 14, -6, 10, 1, 15, -6, 0] Gap of 3

Step 1. [15, 14, -6, 10, 1, 15, -6, 0]

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

Insertion sort each sub array:

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

Combine into one Array by taking the smallest in each sub array.

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

Step 2: Reduce gap by 1. Gap = 2

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

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

Insertion Sort sub arrays:

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

Combine into one array:

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

Step 3: Gap = 1

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

With gap of 1 we use insertion sort to do the rest.

[0, -6, -6, 10, 1, 14, 15, 15] $-6 < 0$

[-6, 0, -6, 10, 1, 14, 15, 15] $-6 < 0$

[-6, -6, 0, 10, 1, 14, 15, 15] $1 < 10$

[-6, -6, 0, 1, 10, 14, 15, 15] Array sorted.

5 (text) [4 points] Rank the six sorting algorithms in order of how you expect their runtimes to compare (fastest to slowest). Your ranking should be based on the asymptotic analysis of the algorithms. You may predict a tie if you think that multiple algorithms have the same runtime. Provide a brief justification for your ranking.

MergeSort: $O(n \log n)$ $\Omega(n \log n)$ 1st
Worst Case: list is in reverse or decreasing order.
Best Case: Already Sorted.

Shell sort: $O(n^2)$ $\Omega(n \log n)$ n = size of list. 2nd
Worst case: list in reverse or decreasing order.
Best Case: Already Sorted.

Quick Sort: $O(n^2)$ $\Omega(n \log n)$ 3rd
Worst Case: The pivot chosen is always the larges or smallest
Best Case: The pivot splits the array into two, equal parts.

Bubble Sort: $O(n^2)$ $\Omega(n)$ n = size of list 4th
Worst case is when list is reverse or decreasing order
Best case: list is already sorted.

Insertion Sort: $O(n^2)$ $\Omega(n)$ n = size of list 4th
Worst Case: list is reverse or decreasing order.
Best case: list is sorted

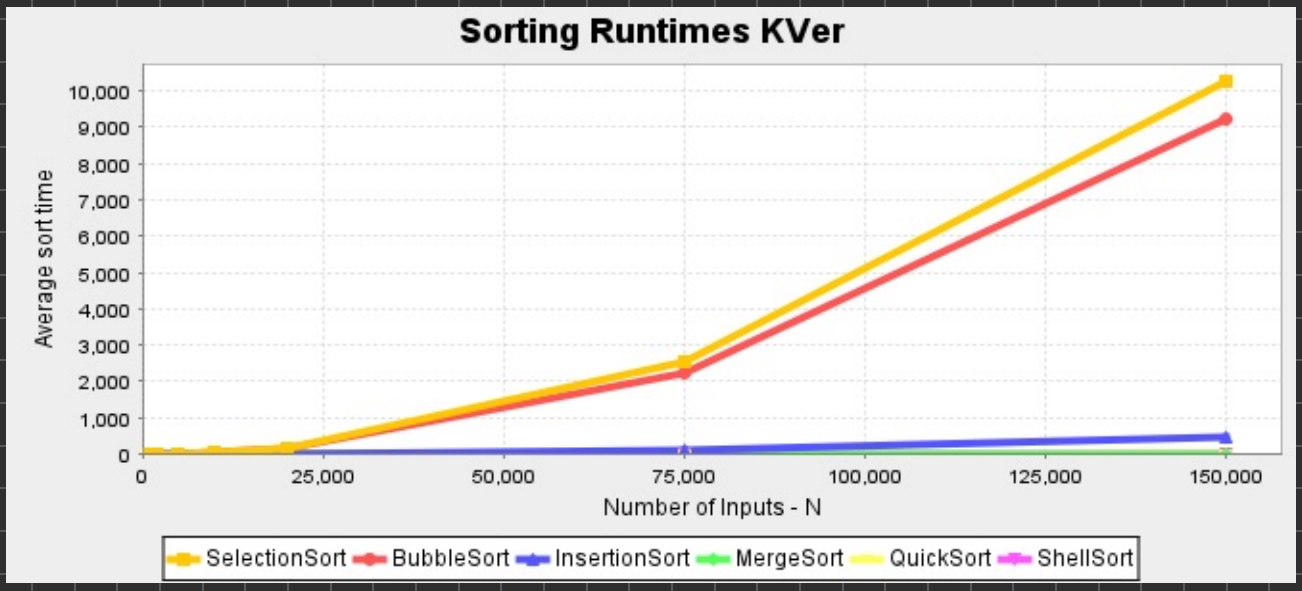
Selection Sort: $O(n^2)$ $\Omega(n^2)$ n = size of list 5th
Worst case: list is reverse or decreasing order. n number of swaps.
Best case: Already sorted. No swapss.

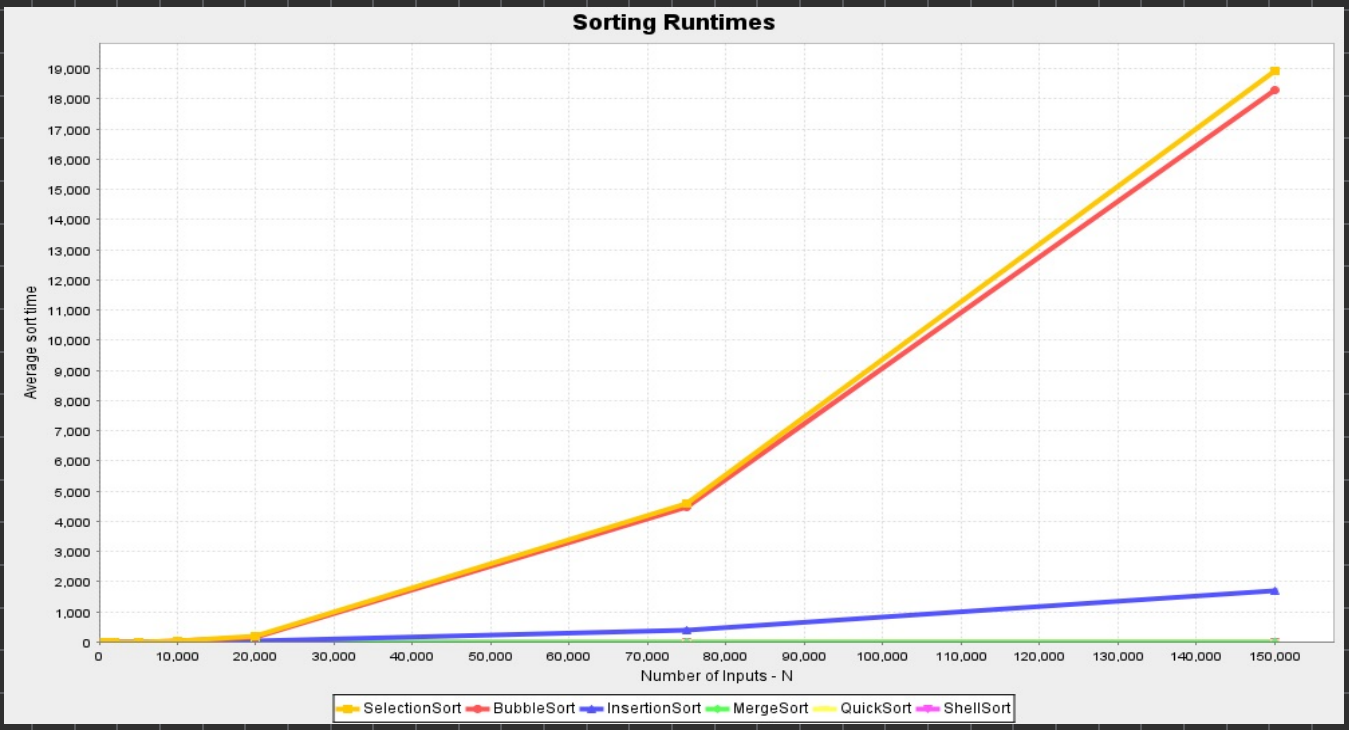
Based on the time complexity, I assume that merge sort will be the best.

I think Shell sort will be faster than quick sort because the worst case for quick sort is more likely to happen. The chance for the list to be reverse sorted is very low even with small input sizes.

Bubble and Insertion sort is a tie because their time complexity is the same.

9 (text) [7 points] Make a graph comparing the runtimes of the six algorithms. Your x-axis should be N (the number of elements you are sorting) and your y-axis should be the average time to sort an array of size N, in milliseconds. You may create this graph using Excel or any other tool of your choice. However, making the chart directly from your java code will award you extra credit. **Note that:** You will find an issue when trying to plot very large numbers (This is on purpose). Once you get this issue, explain what it means and try to fix it to the best of your ability.





10 (text) [7 points] Write a short summary of what you observed in your experiments. What did you notice about the relative performance of the different algorithms? Did the actual performance always match up to the ranking you predicted in part (1) based on the asymptotic analysis? Why or why not?

I noticed that most of the algorithms had similar averages. The ones with the same time complexities have insignificant time differences. I'd say my ranking was accurate, but I did not actually expect for most of times to be similar.