1.) Merge Sort of 2 sorted lists

Compare Results

1 + 3 → [-3]

1 + 0 → [-3, 0]

1 + 16 → [-3, 0, 1]

25 + 16 → [-3, 0, 1, 16]

25 + 16 → [-3, 0, 1, 16, 16]

25 + 27 → [-3, 0, 1, 16, 16, 25]

31 + 27 → [-3, 0, 1, 16, 16, 25, 27]

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

2.) Insertion Sort

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

[-5, -1]

[-5, -1, 67] -10

[-10, -5, -1, 67] 21

[-10, -5, -1, 21, 67] 8

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

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

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

3.) Quicksort

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

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

[-5] [-3] [6]

[-5, -3, 6]

[19] [25] [26, 42]

[26] [42]

[19, 25, 26, 42]

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

4.) Shell Sort

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

4th element [15, 14, -6, 10, 1, 15, -6, 0]

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

2nd element [1, 14, -6, 10, 15, 15, -6, 10]

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

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

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

Question 5.) In order from fastest to slowest:

1. Merge Sort: O(n log n) for best and worst case. – Most consistent
2. Quick Sort – This is generally fast
    a. Best case: O(n log n)
    b. Worst case: O(n^2)
3. Shell Sort: Depends on gap sequence – Typically faster than insertion sort
    a. Best case: O(n log^2 n)
    b. Worst case: O(n^2)
4. Insertion Sort: O(n^2)
    a. Good for small lists or lists that are already almost sorted.
5. Selection Sort: O(n^2)
    a. Very simple to implement, but not the most efficient
6. Bubble Sort: O(n^2)
    a. Bubble sort is typically inefficient for large lists, which is why I put it at the bottom.

Algorithms with lower time complexity like O(n log n) are generally faster especially as the input size grows. Because of this, I think Merge Sort will be the fastest and Bubble Sort will be the slowest.

Question 10.)

Overall Avg Ranking	Name	Time (in ms)
1	Quick Sort	15.6586
2	Merge Sort	23.8801
3	Shell Sort	29.4769
4	Insertion Sort	1992.5651
5	Selection Sort	11252.7703
6	Bubble Sort	31501.3956

The only thing different between my initial ranking in Q5 and here is that Quick Sort ended up being faster than Merge Sort. The algorithms that had time complexities of O(n log n) were the fastest. Insertion sort, selection sort, and bubble sort were much slower with our larger lists. Bubble sort was painfully slow, which was expected since it is inefficient for larger lists. Overall, I am not surprised that Quick Sort ended up outperforming Merge Sort. Even though Quick Sort has a worst case of O(n^2), this is often rare. Merge sort is just more consistent with its time complexity which is why I had it ranked higher.