Keith DeSantis

CS2223 Algorithms

Homework 1 Question 5:

Bubblesort compares two adjacent elements of a collection, from left to right, swaps them if they are out of order, then moves to the next pair. For example, if given [3, 1, 5, 2], it would look at 3 and 1, swap them giving, [1, 3, 5, 2], then compare 3 and 5, then compare 5 and 2, and swap, giving [1, 3, 2, 5]. It repeats this process until no swaps are needed.

Mergesort sorts two halves of a collection, then merges them to be one fully ordered array. It does this through recursion. So, with an input of [3, 4, 2, 1, 7, 6], it would separate into [3, 4, 2] and [1, 7, 6], then recursively sort those to become [2, 3, 4] and [1, 6, 7] and merge those two to finally return [1, 2, 3, 4, 6, 7].

Quicksort determines a partition element, and through a partition function places it in its "sorted position" in the collection, so all elements to its left are less than it and all to its right are greater than it, then recursively sorts those two sub-collections.

Number of steps for each sorting algorithm for varying input sizes (3 repetitions each):

	100	1,000	10,000	100,000	250,000
Bubblesort	7623, 9009, 9108	965034, 977022, 961038	99040095, 99150084, 99410058	1381465693, 1337366134, 1306666441	62418250326, 62377000491, 62323750704

Mergesort	541, 548, 544	8702, 8701, 8698	120385, 120505, 120487	1535935, 1536179, 1536014	4167864, 4167895, 4167590
Quicksort	814, 701, 799	11330, 10820, 12639	153723, 155822, 158226	1971498, 1885614, 1863719	5176525, 5140352, 4926924

Average Steps for each sorting algorithm for varying input sizes:

	100	1,000	10,000	100,000	250,000
Bubblesort	8580	967698	99200079	1341832756	62373000507
Mergesort	544.333	8700.33	120459	1536042.666	4167783
Quicksort	771.333	11596.3	155924	1906943.6666	5081267.333

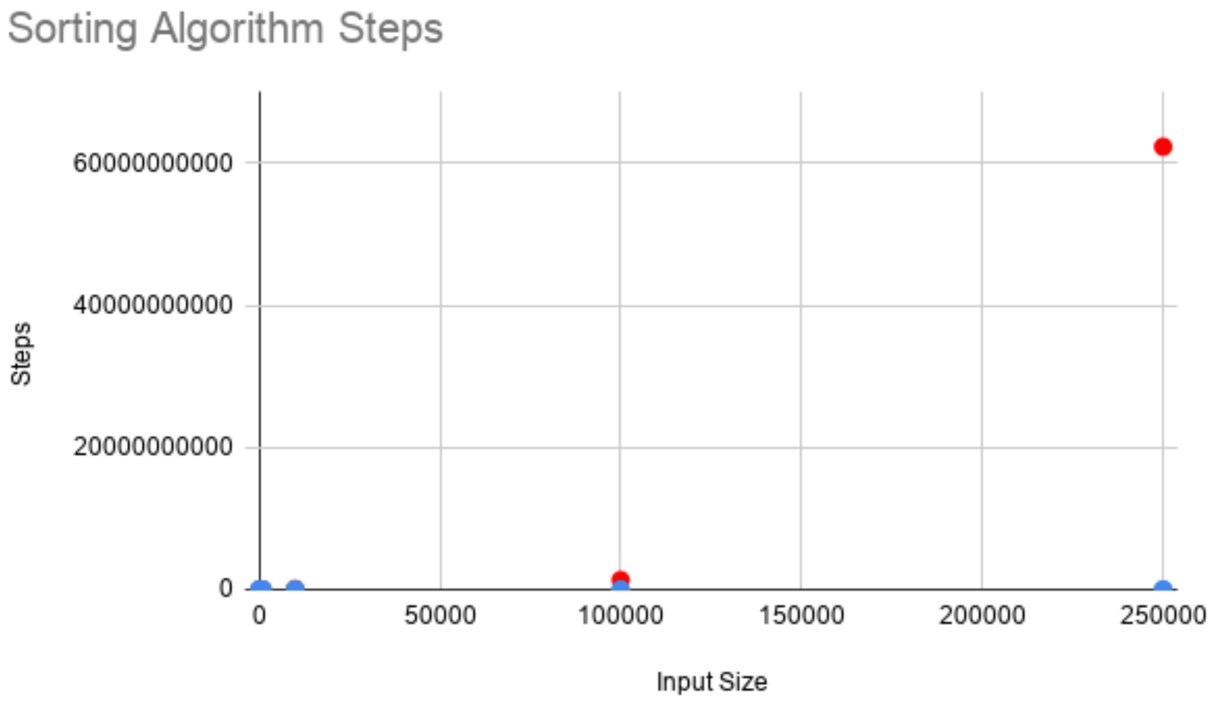

Figure 1. shows the results of the trials, with Bubblesort (red), Mergesort (green), and Quicksort (blue). While it is clear Bubblesort was the least efficient, a log scale is necessary to get better visualization.

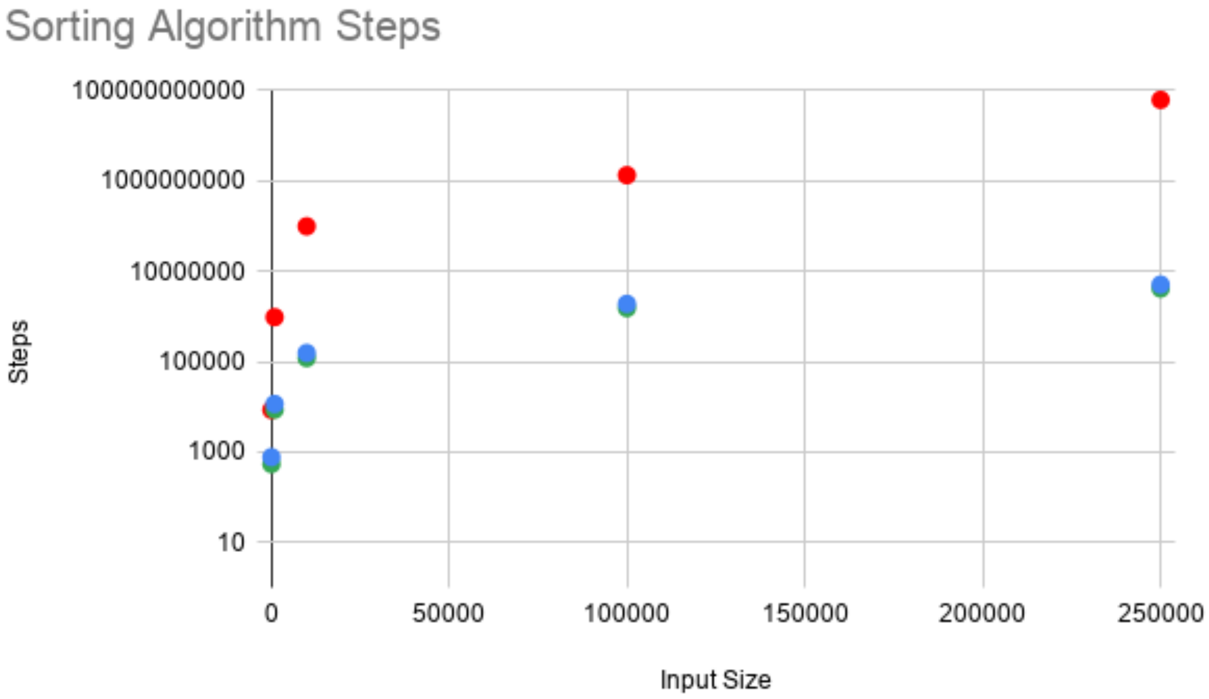

Figure 2. shows the results graphed with a log scale, with Bubblesort (red), Mergesort (green), and Quicksort (blue).

Bubblesort was by far the least efficient sorting algorithm, operating orders of magnitude above the others, Mergesort was consistently faster than Quicksort, seemingly not by an order of magnitude but rather by a constant factor. Overall, Mergesort and Quicksort were effective and efficient algorithms.