

Shell sort appeared to have outperformed Bubble Sort, Selection Sort, and Insertion Sort for *most* cases, except when the file was already sorted. Shell Sort would then perform unnecessary comparisons. Quick sort had similar comparisons to the three original algorithms, however had considerably less in the case of a randomly sorted file. Quick sort also had less exchanges than bubble sort and insertion sort in the case of a randomly sorted file, and a file that had to be sorted backwards when it was already sorted.

Quick Sort’s worst case appeared to be when the file was already sorted, since it would make unnecessary exchanges and comparisons. Although it had a bit more exchanges in the case of a randomly sorted file, in the case of the file either being completely sorted or sorted backwards, quick sort had considerably more comparisons than if it was randomly sorted. When it’s already sorted, it goes until the end without making an exchange. In the case of it having to sort it when it’s sorted backward, the algorithm has to go through and compare/exchange each element.

Shell sort’s worst case appeared to be when the file was sorted randomly. Although in all three cases (random sort, already sorted, and sorting backwards), the comparisons were *similar*, it appeared to be that a completely sorted file was the best case, since it had the least comparisons, and didn’t make any comparisons. Contrast to quick sort, if there’s no need to make an exchange, the algorithm will not make an exchange.

The two best incremental sequences for Shell Sort appeared to be the sequences 2\*h+1 and 3\*h+1. Both on average had less comparisons and exchanges than the sequence 4\*h+1. If the file was completely sorted, 4\*h+1 would be the best sequence, since it had the least comparisons and (like the other sequences of course) had 0 exchanges. This sequence also had less comparisons for if the data was to sort an already sorted file backwards, however it had the most exchanges. The sequence 2\*h+1 performed better than 3\*h+1 in terms of exchanges, however the sequence 3\*h+1 performed better in terms of comparisons than 2\*h+1.