Algorithm Analysis

The most effective sorting algorithm seems to depend on the starting order of the data array to be sorted. For a randomly filled array, the Median of Three sorts seemed to have the lowest average run-time with little difference being made by the base case array size. Simple Quicksort, Random Pivot and MergeSort were all less efficient than the Median of Three sorts, while the Insertion Sort and Shell Sort were clearly the most inefficient of all the algorithms.

For a data array that is already sorted, the Insertion Sort, Shell Sort, and MergeSort algorithms were clearly the most efficient of all of them, with the Insertion Sort and Shell Sort both being slightly more efficient than the MergeSort. The Median of Three sorts were the next most efficient, and seemed to become less efficient as the base case array size decreased. The Random Pivot sort was not very efficient, but the Simple Quicksort was the least efficient in the case of a presorted array.

For a data array that is reverse sorted, the Median of Three sorts were the most efficient sorting algorithms. The base case array size of 100 was the most efficient of the three Median of Three sorts. Random Pivot was the next most efficient followed by the MergeSort. The Simple Quicksort was not very efficient at sorting the reverse sorted arrays, but the Insertion Sort and ShellSort were the least efficient of all of them.

Overall, I would say the Median of Three sorts are the most efficient for sorting a data array. While the MergeSort was able to recognize a presorted array faster than the Median of Three sorts, it was shown to be much less efficient with a randomly sorted data array and a reverse sorted data array. It also seems that the Median of Three sort is more efficient with a larger base case array size, but in randomly sorted data array the base case array size does not seem to play as big of a role in average runtime.