Michael Tardi

Sorting Final Project

Of the sorting algorithms Quick sort would be the fastest. As its name implies, quick sort sorts each value the quickest. By making a pivot it sorts the values either left or right of the pivot depending on whether or not the value in the array is greater than or less than the pivot. By doing so it allows the values to be sorted partially before fully sorting each value. The result of this is the speed at which the sorting is done. The next fasted sort would be merge sort. As quick sort passes values through a pivot, merge sort is similar to quick sort as it separates values into multiple arrays and then sorts them when they because individual numbers. This sorting algorithm is very efficient as it can compare to quicksort in terms of speed. While quicksort is fast at smaller arrays merge sort beats quicksort at larger ones. While quicksort has to sort by a pivot because of this it takes an extra step in its process of sorting while merge sort can recursively divide and conquer the array resulting in a faster sort with larger arrays. The next fastest sorting algorithm is heapsort. This sort is the next fastest as its big O value is nlogn in its worst case along with merge sort. The result being its sorting speed being quite quick. The next fastest sort would be insertion sort as it beats out the rest of the sorting algorithms based on the timing graphed. This beats out selection by a slim margin on the larger value array but is similar on the smaller value array. Selection sort is next as it’s not much slower than insertion sort but still comes behind insertion sort. Bubble sort comes in next as it is the about the same as insertion and selection when it comes to the ten thousand value array but becomes slower when reaching the higher array of two hundred thousand. Lastly shell sort is the slowest sort at the high values of ten thousand and two hundred thousand. Shown by the graphs it is not much farther behind the other big O(n^2) sorting algorithms but falls behind them at the large values. While it may be faster than insertion sort with smaller value arrays it is seen to be slower than the rest when it comes to the larger value arrays. All in all, the fastest sort is shown to be quick sort, followed by merge sort then heap sort, next would-be insertion sort then selection sort followed by bubble sort, and lastly would be shell sort. These sorting algorithms vary in speed, but can all be used depending on the size of the array. When the size of the array is as large as ten thousand and two hundred thousand it shows which arrays are suitable to use and which arrays fall behind the others in use, when the array size is at smaller and larger values. These array sizes may not be a great example because there are only two and even though they are at significantly different values more may be needed in order to get an accurate measurement of the sorting algorithms.