Scott Fenton

Project 3

**Before Improvements**

Enter the data file name:

[Mystery.txt]

For file mystery.txt of size 26, the number of Selection Sort comparisons is: 325

For file mystery.txt of size 26, the number of Insertion Sort comparisons is: 143

For file mystery.txt of size 26, the number of Bubble Sort comparisons is: 325

For file mystery.txt of size 26, the number of Quick Sort comparisons is: 180

For file mystery.txt of size 26, the number of Merge Sort comparisons is: 84

Max Memory Use = 26

Enter the data file name:

[random1.txt]

For file random1.txt of size 26, the number of Selection Sort comparisons is: 325

For file random1.txt of size 26, the number of Insertion Sort comparisons is: 196

For file random1.txt of size 26, the number of Bubble Sort comparisons is: 325

For file random1.txt of size 26, the number of Quick Sort comparisons is: 194

For file random1.txt of size 26, the number of Merge Sort comparisons is: 92

Max Memory Use = 26

Enter the data file name:

[random2.txt]

For file random2.txt of size 52, the number of Selection Sort comparisons is: 1326

For file random2.txt of size 52, the number of Insertion Sort comparisons is: 638

For file random2.txt of size 52, the number of Bubble Sort comparisons is: 1326

For file random2.txt of size 52, the number of Quick Sort comparisons is: 410

For file random2.txt of size 52, the number of Merge Sort comparisons is: 237

Max Memory Use = 52

Enter the data file name:

[reverse.txt]

For file reverse.txt of size 26, the number of Selection Sort comparisons is: 325

For file reverse.txt of size 26, the number of Insertion Sort comparisons is: 350

For file reverse.txt of size 26, the number of Bubble Sort comparisons is: 325

For file reverse.txt of size 26, the number of Quick Sort comparisons is: 387

For file reverse.txt of size 26, the number of Merge Sort comparisons is: 57

Max Memory Use = 26

Enter the data file name:

[sorted.txt]

For file sorted.txt of size 26, the number of Selection Sort comparisons is: 325

For file sorted.txt of size 26, the number of Insertion Sort comparisons is: 25

For file sorted.txt of size 26, the number of Bubble Sort comparisons is: 325

For file sorted.txt of size 26, the number of Quick Sort comparisons is: 400

For file sorted.txt of size 26, the number of Merge Sort comparisons is: 67

Max Memory Use = 26

**After Improvements**

Enter the data file name:

[mystery.txt]

For file mystery.txt of size 26, the number of Selection Sort comparisons is: 325

For file mystery.txt of size 26, the number of Insertion Sort comparisons is: 143

For file mystery.txt of size 26, the number of Bubble Sort comparisons is: 247

For file mystery.txt of size 26, the number of Quick Sort comparisons is: 347

For file mystery.txt of size 26, the number of Merge Sort comparisons is: 84

Max Memory Use = 13

Do you want to sort another data file?

[random1.txt]

For file random1.txt of size 26, the number of Selection Sort comparisons is: 325

For file random1.txt of size 26, the number of Insertion Sort comparisons is: 196

For file random1.txt of size 26, the number of Bubble Sort comparisons is: 324

For file random1.txt of size 26, the number of Quick Sort comparisons is: 171

For file random1.txt of size 26, the number of Merge Sort comparisons is: 92

Max Memory Use = 13

Do you want to sort another data file?

[random2.txt]

For file random2.txt of size 52, the number of Selection Sort comparisons is: 1326

For file random2.txt of size 52, the number of Insertion Sort comparisons is: 638

For file random2.txt of size 52, the number of Bubble Sort comparisons is: 1281

For file random2.txt of size 52, the number of Quick Sort comparisons is: 574

For file random2.txt of size 52, the number of Merge Sort comparisons is: 237

Max Memory Use = 26

Do you want to sort another data file?

[reverse.txt]

For file reverse.txt of size 26, the number of Selection Sort comparisons is: 325

For file reverse.txt of size 26, the number of Insertion Sort comparisons is: 350

For file reverse.txt of size 26, the number of Bubble Sort comparisons is: 325

For file reverse.txt of size 26, the number of Quick Sort comparisons is: 170

For file reverse.txt of size 26, the number of Merge Sort comparisons is: 57

Max Memory Use = 13

Enter the data file name:

[sorted.txt]

For file sorted.txt of size 26, the number of Selection Sort comparisons is: 325

For file sorted.txt of size 26, the number of Insertion Sort comparisons is: 25

For file sorted.txt of size 26, the number of Bubble Sort comparisons is: 25

For file sorted.txt of size 26, the number of Quick Sort comparisons is: 153

For file sorted.txt of size 26, the number of Merge Sort comparisons is: 67

Max Memory Use = 13

1. The first algorithm that was changed in the SortingAndSearching class was the bubble sort method. In this method I changed the code so that it kept track of no swaps taking place, which meant that if no swaps happened then it must be already in sorted order. We can see this reflected in the baseline report for sorted.txt file. The bubble checks if the values are in their proper place for every single word, even if they are already sorted, resulting O(n^2) comparisons. In the improved version, we stops the sorting algorithm when we see that no more swaps are taking place, this results in a O(n) comparisons for the already sorted data, and the maximum O(n^2) comparisons for data sorted in reverse order. This also results in slightly less comparisons in the improved version of mystery.txt,random1.txt, and random2.txt because it can skip a few comparisons toward the end of the sorting if the file last few words are already in the correct place.
2. The change in the quicksort method was to change the pivot from the first element to the middle element. The primary reason to change the pivot was to decrease the comparisons made if the data was already sorted, or in reverse order. If that data is already sorted or reverse, then having the pivot at the first element causes worst case behavior of O(n^2). With the improvement we increased the baseline from O(n^2) to O(n log(n)). This is a large improvement for sorted and reverse sorted data because these are common type of data you can get. The mystery.txt, random1.txt, and random2.txt either went up slightly in comparisons from baseline or down slightly. This is because the pivot has an effect on the comparisons depending on the level of sorting the data is already in.
3. The difference in the memory count from the baseline to the improved version is 1/2N. This is apparent because the change in the merge sort allocates the size of the array after the recursive call, meaning that the largest N value can only be ½ the size of the array. This means for a text file with 26 words, the merge sort would only need to allocate a size 13 array, and a file with 52 words would only need to allocate a size of 26 array. The final array has to be either 26 or 52 because you end up merging the two half’s, but the temporary arrays only need to be a maximum size of 13 or 26 respectively.
4. The mystery.txt file was determined to be a file that is not in any sorted order, similar to random1.txt and random2.txt. We can see that this is the case because it does not give best case scenario with the quicksort method that would result in O(n), but it also does not have worst case scenario with Insertion sort O(n^2), that we would see if it was sorted in reverse order. We can see that the baseline is quite similar between mystery.txt, random1.txt, and random2.txt, with similar Big-o notation for all of the sorting methods used. An interesting problem occurred with sorting the mystery.txt with the quicksort method. This caused worse case behavior of O(n^2), which can be expected if we choose the middle pivot. The reason for the worst case behavior is that more often than not, data is either sorted or in reverse order, meaning that choosing the middle pivot is best practice. However, in the case of the mystery file, since the file is in random order choosing the middle pivot results in increased comparisons compared to choosing the first element or last element. Had we chosen the first element as in the baseline case, we would have seen O(n log(n)) behavior.