Design & Analysis of Algorithms

Assignment #1

UIN: 815280798

Name: Logan Esala

09/19/2025

6.5: Report and analysis

1.

I implemented the insertion, and merge sort in C++ using the visual studio IDE,

I used vectors to store the data for the testing and created 4 files to implement the entirety of the assignment. I implemented a top-down recursive merge sort and a standard iterative insertion sort for this assignment.

The insertion sort works by building a sorted list one element at a time and it shifts elements over into the correct position by comparison.

The merge sort uses a divide and conquer strategy breaking the data into small half sub arrays in a recursive way, the data are sorted and then the arrays are merged back together.

2.

Time complexity Analysis

The insertion sort I implemented has a best-case of O(n) a worst-case of O(n^2) and an average case of O(n^2)

The merge sort on the other hand has a time complexity of O(n log n) for all cases

The equation for the merge sort process is: $T(n) = 2T(n/2) + O(n)$

Where T(n) = time to sort n elements

2T(n/2) = time for two recursive calls (left + right halves)

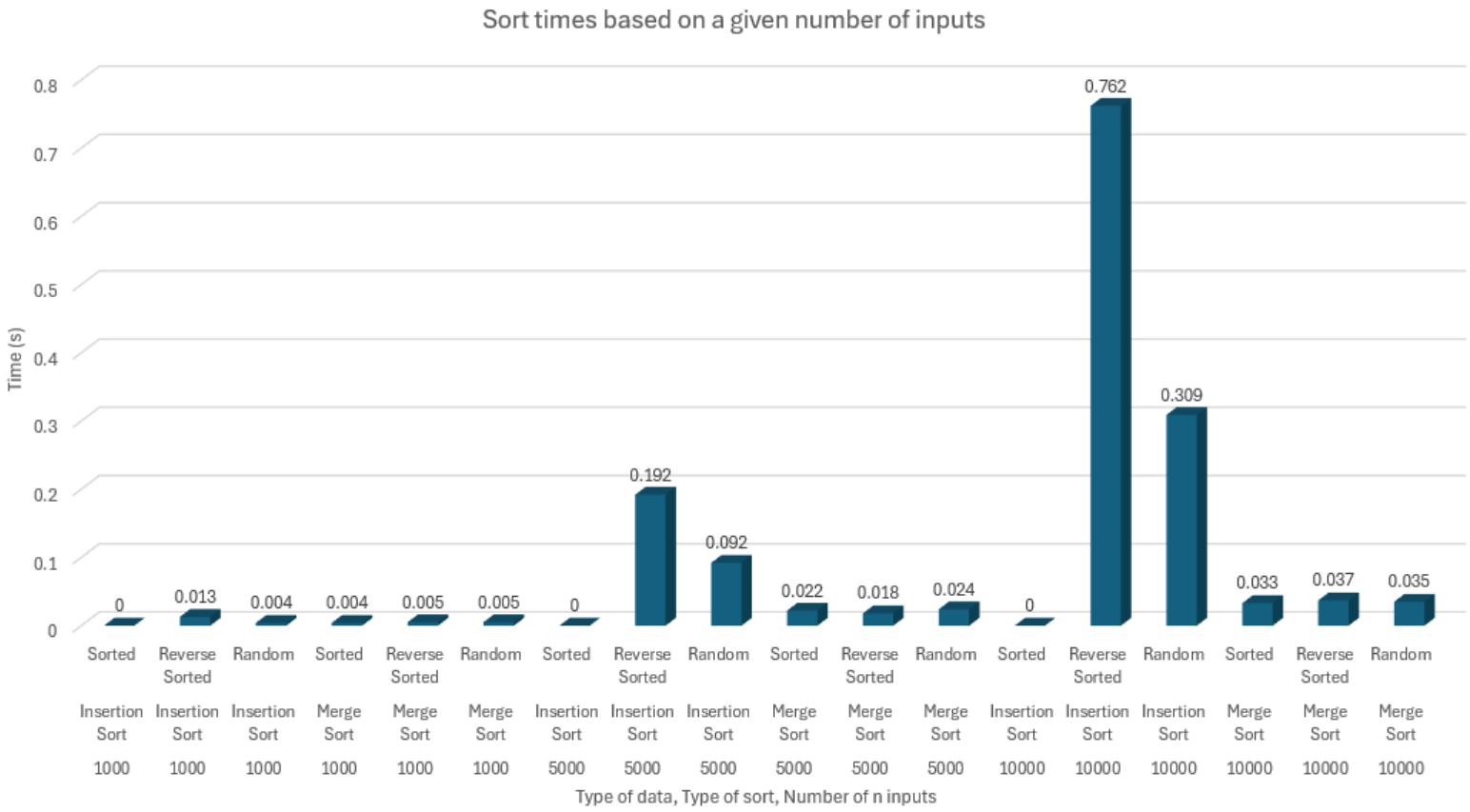
O(n) = time to merge the two halves

Comparing to the master theorem cases we get n log n applying to the setup.

```
Insertion Sort with sorted data 1000 took 0 seconds
Insertion Sort with reverse sorted data 1000 took 0.007 seconds
Insertion Sort with Randomly generated data 1000 took 0.004 seconds
Merge Sort with sorted data 1000 took 0.004 seconds
Merge Sort with reverse sorted data 1000 took 0.006 seconds
Merge Sort with Randomly generated data 1000 took 0.006 seconds
Insertion Sort with sorted data 5000 took 0 seconds
Insertion Sort with reverse sorted data 5000 took 0.162 seconds
Insertion Sort with Randomly generated data 5000 took 0.121 seconds
Merge Sort with sorted data 5000 took 0.02 seconds
Merge Sort with reverse sorted data 5000 took 0.021 seconds
Merge Sort with Randomly generated data 5000 took 0.017 seconds
Insertion Sort with sorted data 10000 took 0.001 seconds
Insertion Sort with reverse sorted data 10000 took 0.76 seconds
Insertion Sort with Randomly generated data 10000 took 0.317 seconds
Merge Sort with sorted data 10000 took 0.033 seconds
Merge Sort with reverse sorted data 10000 took 0.032 seconds
Merge Sort with Randomly generated data 10000 took 0.033 seconds
```

3.

Experimental Results:



4.

Observations & Conclusions

At the beginning of the test the Insertion sort had a slight advantage because the sorted data allowed for it to achieve a best case time of O(n) while Merge sorts divide and conquer took longer with the smaller sorted data because it has to perform the breaking up of the inputs regardless. The Merge sort performed better on most occasions but performed much better than the insertion sort as the n number of inputs grew. Merge sort is a fast n log n sorting algorithm and breaks up data into smaller sub arrays that are sorted, this divide and conquer process allows merge sort to out compete the insertion sort algorithm as the number of inputs increases. The drawback of this is that the speed comes at the cost of memory, creating new sub arrays to store fractions of the original array creates a memory cost because as the number of inputs grows the memory usage does as well because more and more sub arrays need to be created to house the extra inputs.