Kwasi Osae-Kwapong HW1 3/5/21

The following aims to compare the runtime of three algorithms including a naive implementation of insertion sort, an improved implementation of insertion sort and an implementation of merge sort. To compare them each algorithm was tested using matrix inputs with dimensions M, the size of the random integer vector array, and N, the dimensions of the integer vector array. The matrix input can be in any of three directions, sorted, representing the best case, randomly sorted, representing the average case, and reverse sorted, representing the worst case. I’ve found that the naive implementation performs as expected in time and is sensitive to the directionality of the input matrix. The improved implementation of insertion sort differs from the naive implementation in that it calculated the lengths of the vectors before any comparisons are made. This difference in the algorithm does not seem to improve runtime, however, it does change how the algorithm performs when the directionality of the input is changed. The merge sort implementation notably fails to handle inputs with N values greater than 1, which implies that the implemented algorithm is not suitable for matrices.

The naive insertion sort runtime is significantly affected by changes in M, N and the direction of the input. Figure 1 shows the runtime of the three input directions run ten times with an M=10,000 and N =10. As anticipated the reverse ordered input has the longest runtime on average and represents the worst case scenario.

*Figure 1: Naive Insertion Sort Input Direction Comparison*

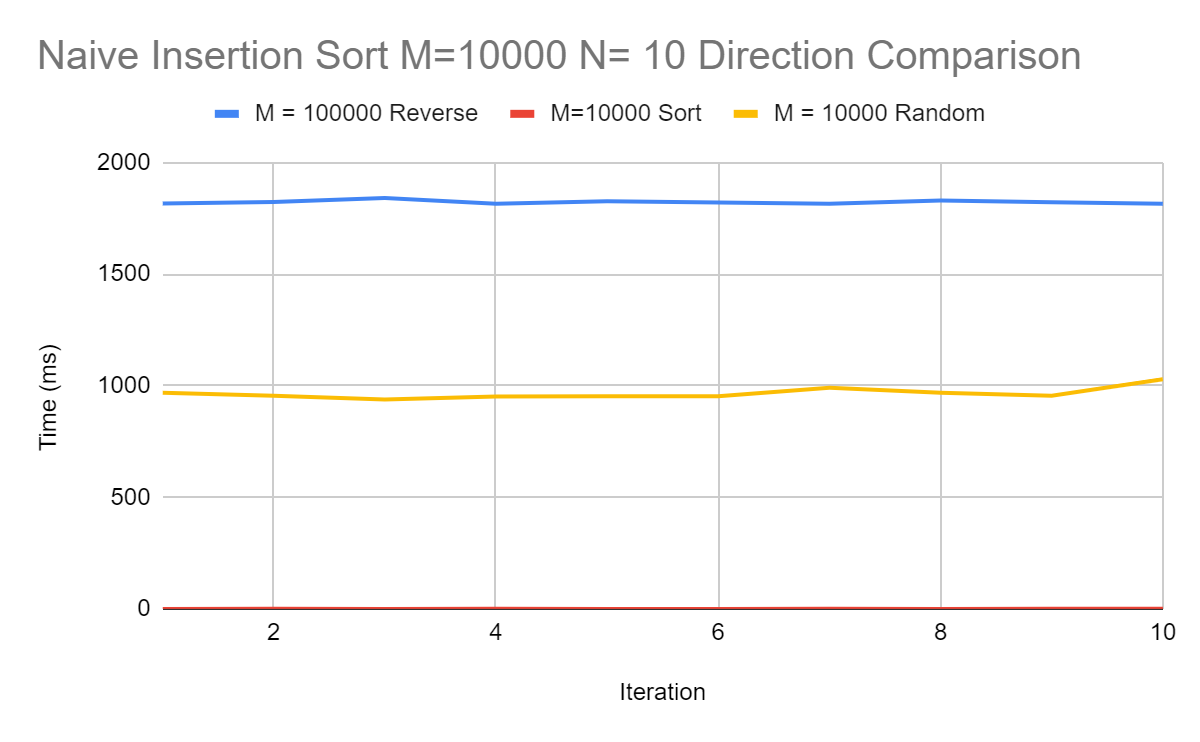
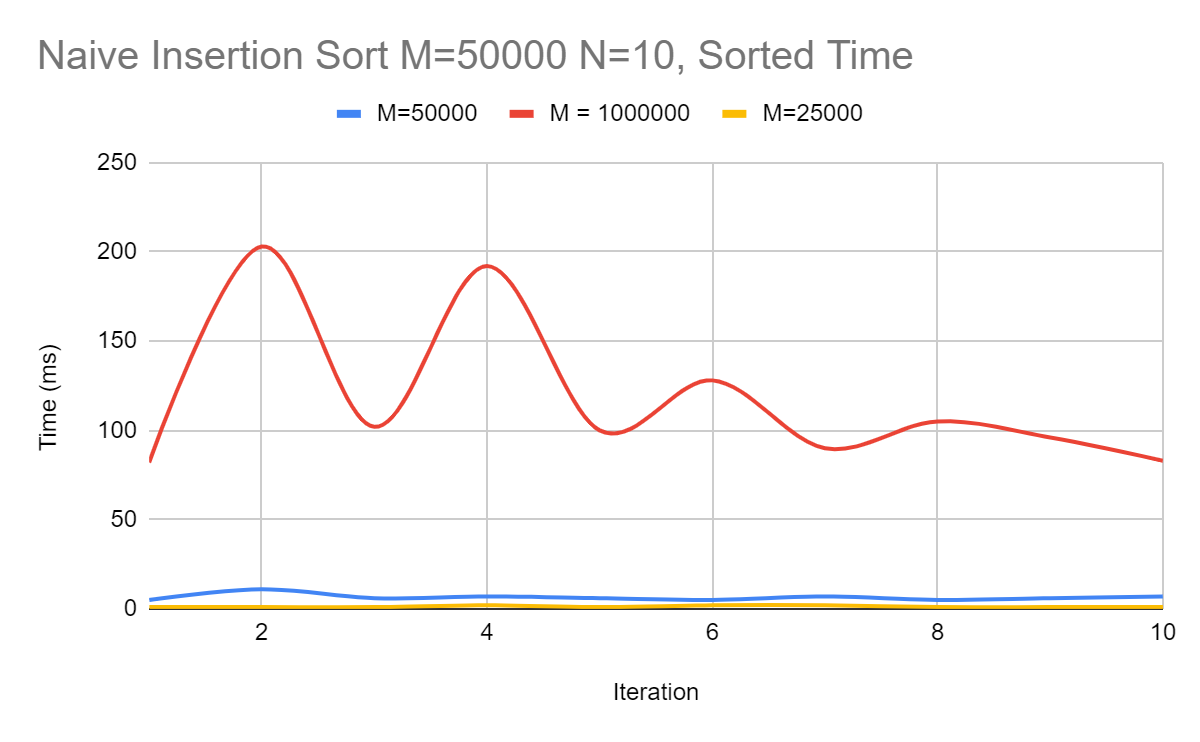


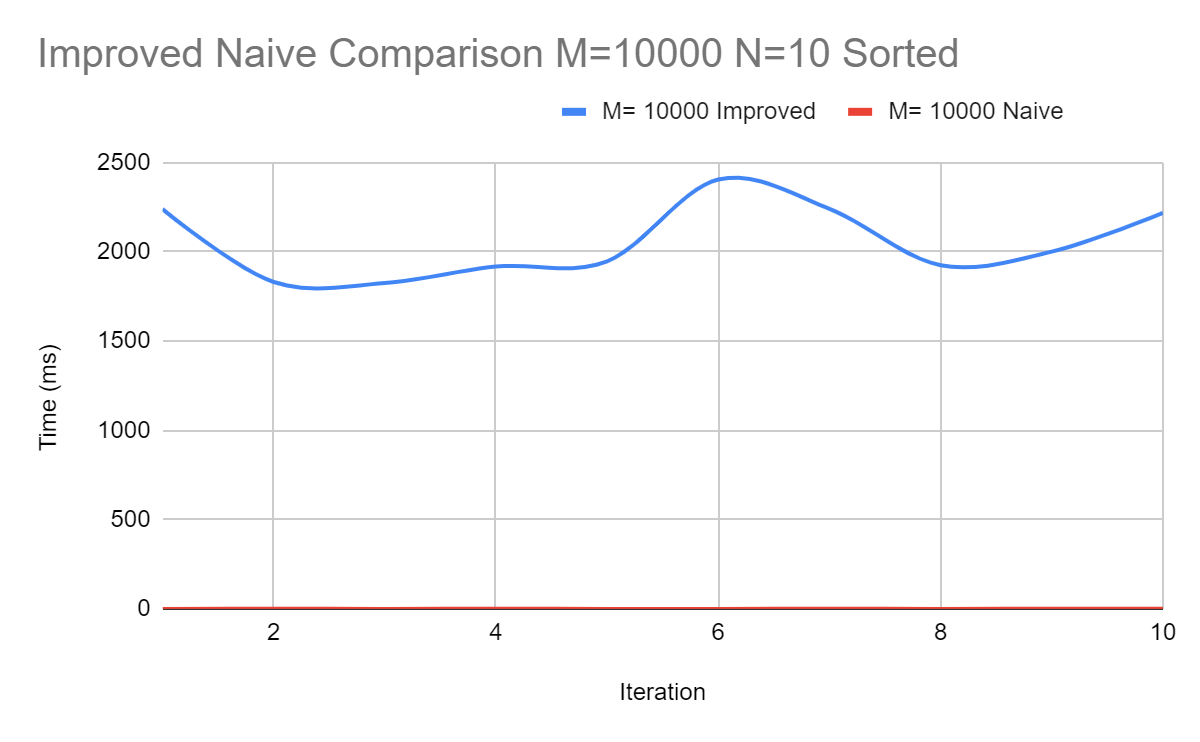
Figure 2 shows the consequences of increasing M on the naive implementation. Ten iterations of sorting were run with M values of 25000, 50000, and 1000000 and an N value of 10. As input size increases so does the runtime. As expected, the naive version is quite sensitive to the input integer vector array.

*Figure 2: Naive Insertion Sort with Varying M Values Sorted*



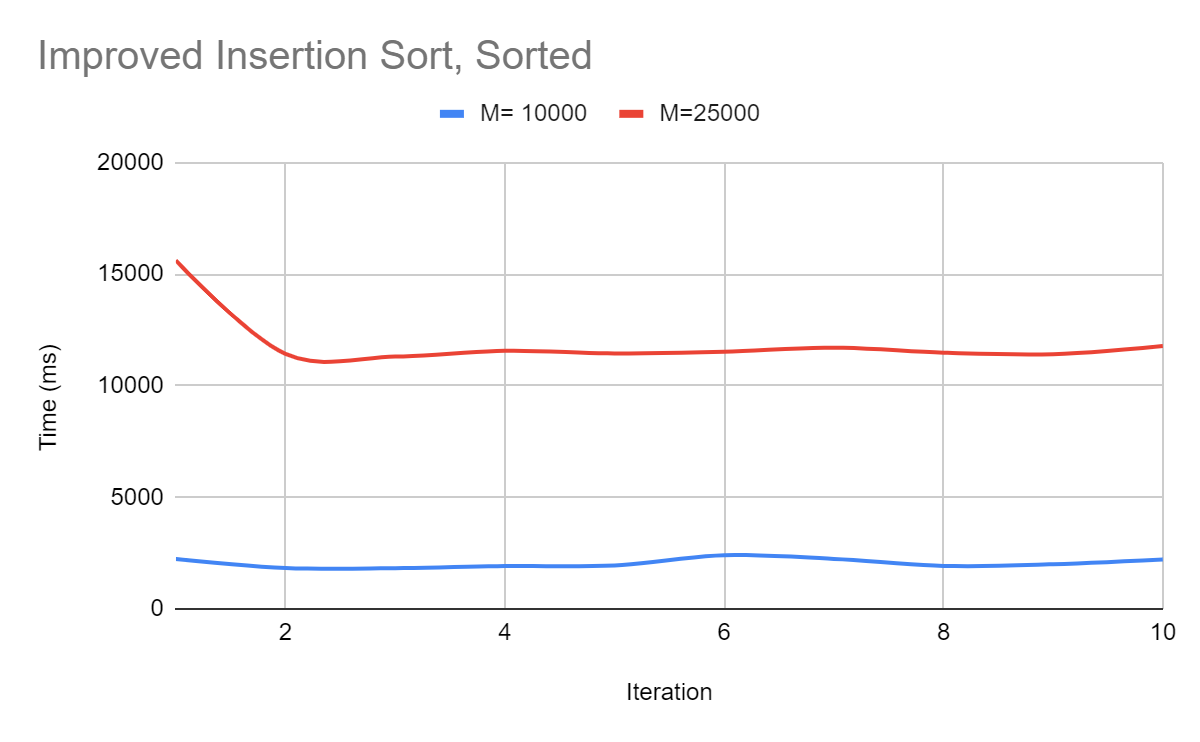
Comparing the naive insertion sort to the improved implementation of insertion sort, some peculiar observations are made. First, when comparing with a constant M value the improved implementation runs significantly slower than it’s naive counterpart. Figure 3 demonstrates this unexpected result.

*Figure 3: Improved Compared to Naive Implementation*



Unfortunately, my “improved” implementation of insertion sort performs much worse than expected when sorting an input vector in any direction. This implementation of insertion sort is also more sensitive to changes in M value compared to the naive version. Figure 3 shows the ten fold increase in runtime when the input size goes from M = 10000 to M= 25000. The naive version does not experience such a significant increase in runtime with increasing M value.

*Figure 3: Improved Insertion Sort with Varying M Value N= 10 Sorted*



Interestingly, the improved algorithm provides approximately the same average runtime when the input direction is changed. Figure 4 shows the runtimes of the sorted, reversed, and random input direction. This feature may be due to calculating the length of the vectors before the comparisons are made.

*Figure 4: Improved Insertion Sort Input Direction Comparison*

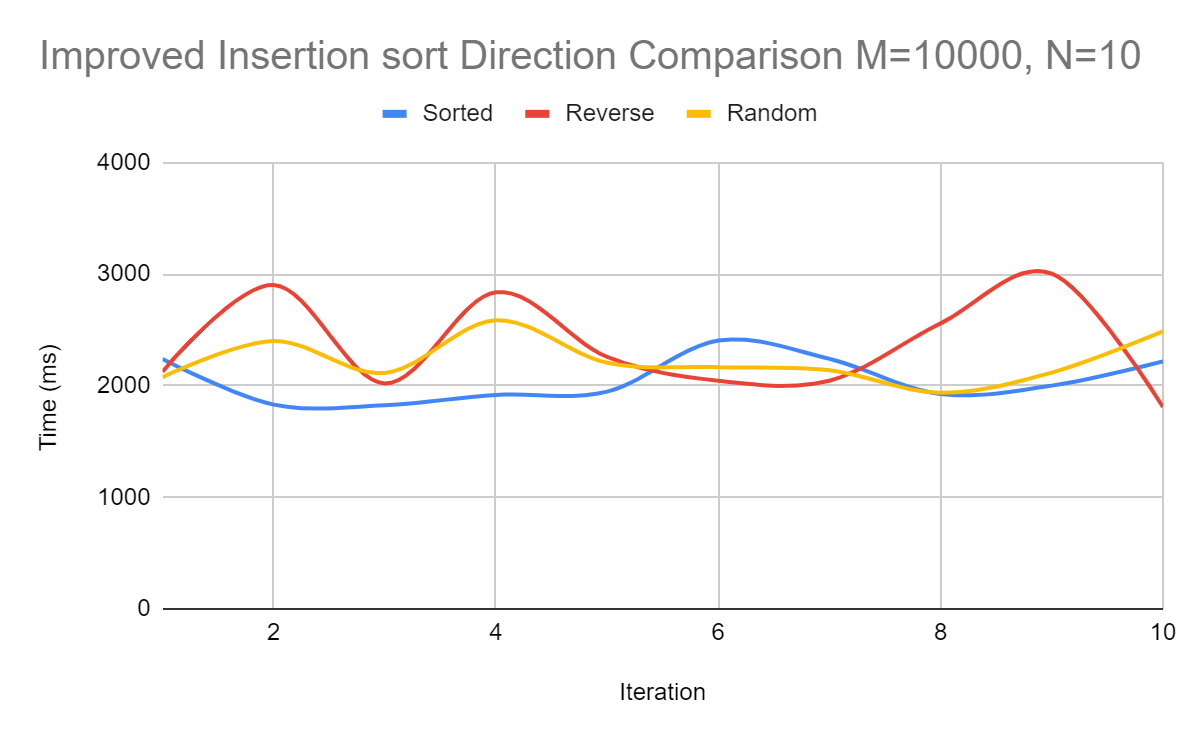
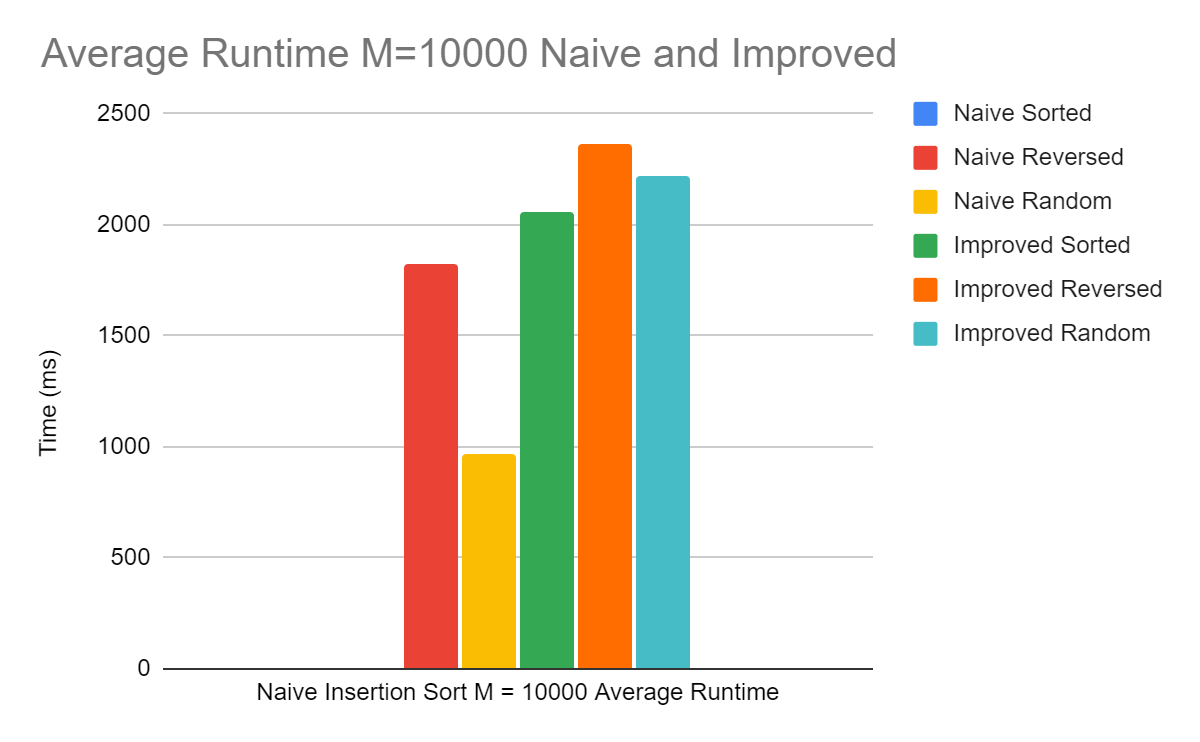


Figure 5 shows the averages runtimes for the naive and improved implementations of insertion sort. It’s clear that on average the improved insertion sort underperforms compared to the naive case

*Figure 5: Average Runtime for M= 10000 Naive and Improved Insertion Sort*



The merge sort implementation presented unique behavior during data collection. Theoretically the implemented code should run, however, when tested against inputs the following exception is thrown “0 [main] hw1 951 cygwin\_exception::open\_stackdumpfile: Dumping stack trace to hw1.exe.stackdump”. I could not find the source of the exception as no valid line number is given. Searches on the internet suggest that it may be due to a function receiving a value when it needs a pointer, however, the implemented code is sound. As a result of this error no data could be collected for merge sort.