1. Merge Sort and Insertion Sort Programs

   Insertion sort pseudo code
   1. Iterate from arr[1] to arr[n] over the array
   2. Compare the current element “key” to its predecessor
   3. If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

   Merge sort pseudo code
   1. If the right array is shorter than the left aray
   2. Find the middle point to divide the array into two halves:
      a. middle m = l + (r - l) / 2
   3. Call mergeSort for the first half:
      a. Call mergeSort(arr, l , m)
   4. Call mergeSort for the second half:
      a. Call mergeSort(arr, m+1, r)
   5. Merge the two halves sorted in steps 2 and 3:
      a. Call merge(arr, l, m, r)

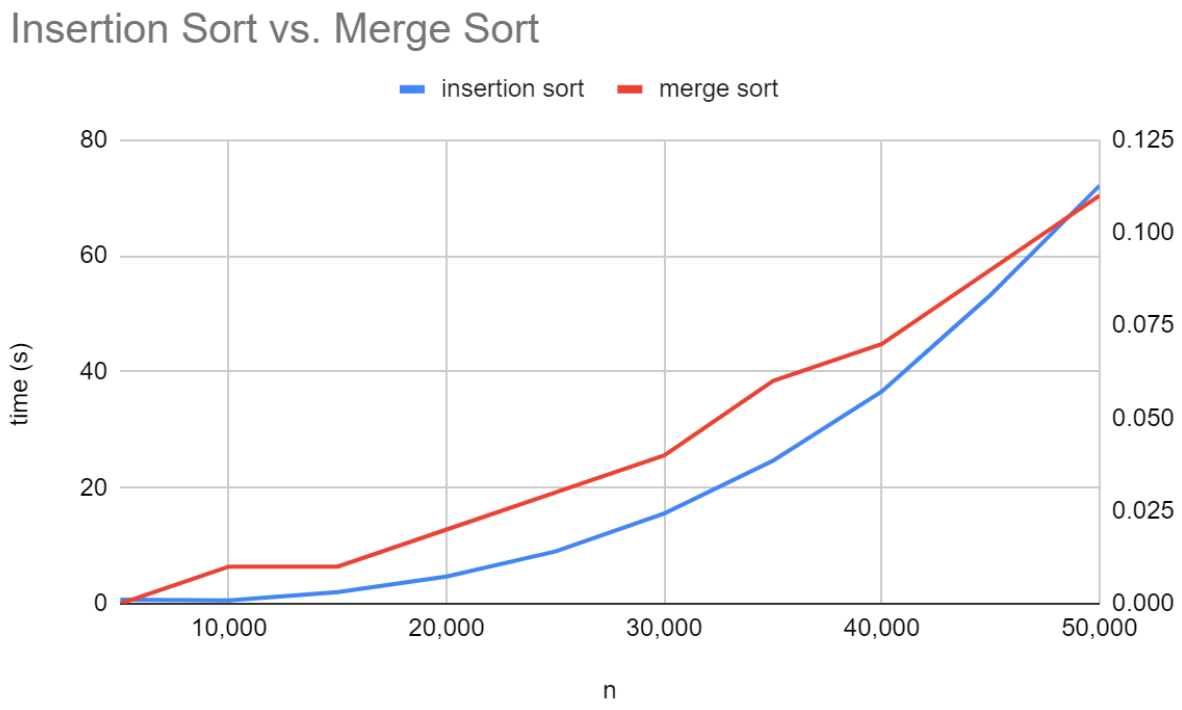
2. Merge Sort vs Insertion Sort Running Time Analysis

   For both the insertTime and mergeTime programs, I modified the program to create vectors of random lengths up to 10,000 integers. I then passed in the randomly generated vectors into the sorting algorithm that had a start and stop timer before and after the algorithm call to later calculate time. All of this was in a for loop that iterates in increments of 5000 up to 50,000. Every 5000 vectors that get sorted, the for loop prints out the time it took to sort. The sorting algorithm itself did not change between the sorting program and the time program because since they worked, they should work for vectors or arrays of any length.

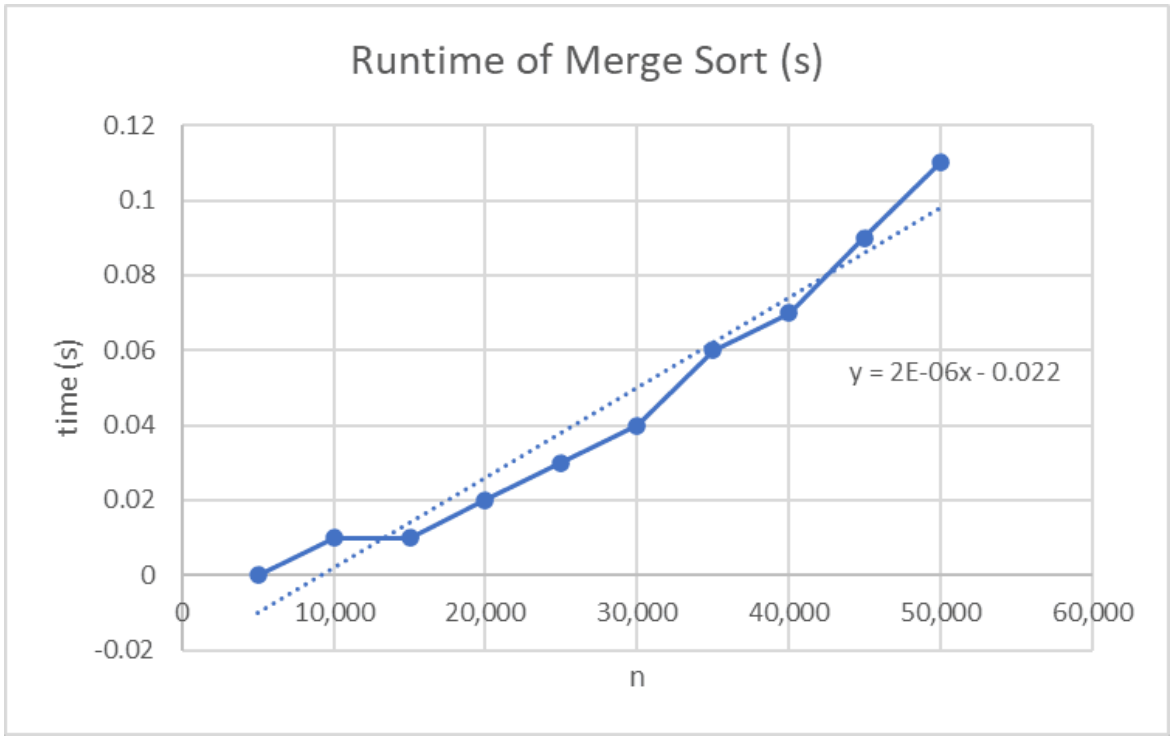
3. Written Report and Data Analysis

Runtime Data

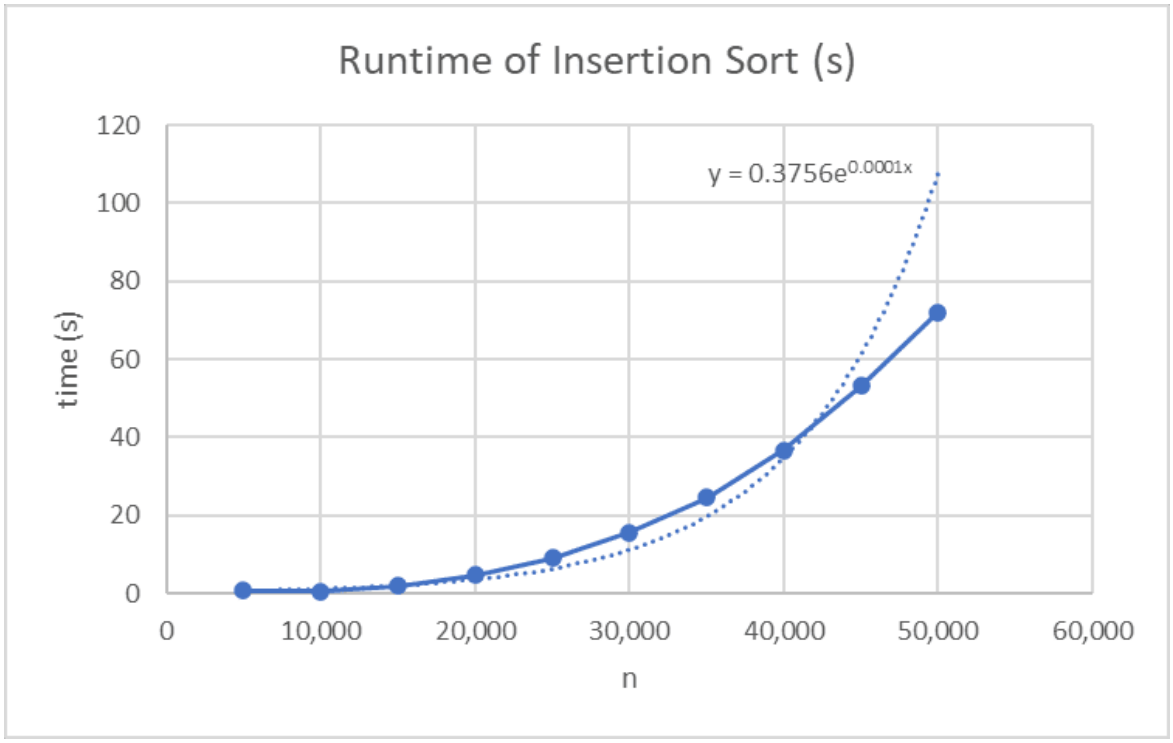
n	time (s)	time (s)
5,000	0	0.7
10,000	0.01	0.58
15,000	0.01	2.01
20,000	0.02	4.67
25,000	0.03	9.01
30,000	0.04	15.57
35,000	0.06	24.65
40,000	0.07	36.61
45,000	0.09	53.26
50,000	0.11	72.09



These are the average run time cases for these sorting algorithms.
Both of the sorting algorithms have similar runtime to their theoretical run times. However, the merge sort algorithm ran significantly faster than the insertion sort.



A curve using O(n(log(n)) best fits the data of merge sort.
Merge sort n = 500,000 time prediction 1.178 seconds



A curve using O(n^2) best fits the data of insertion sort.

Insertion sort n = 500,000 time prediction 738.265 seconds