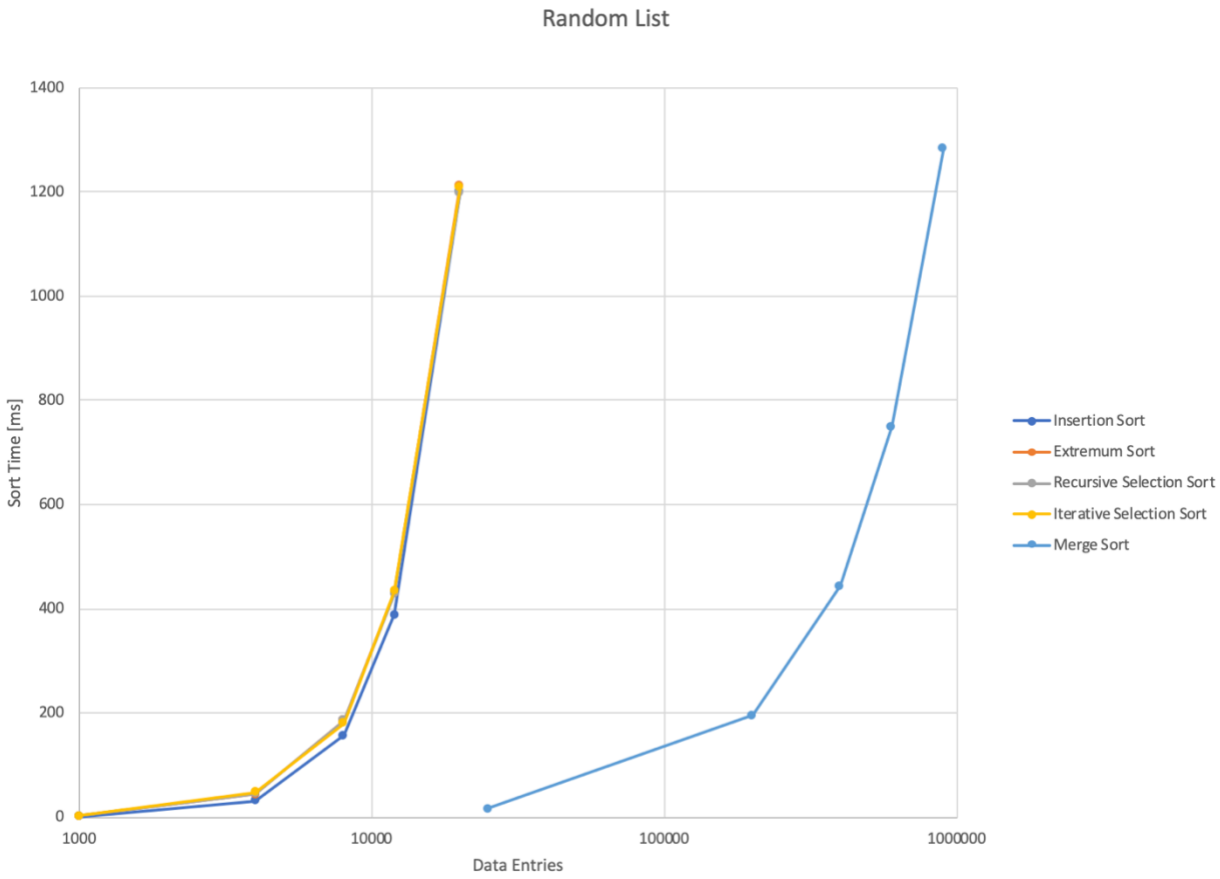
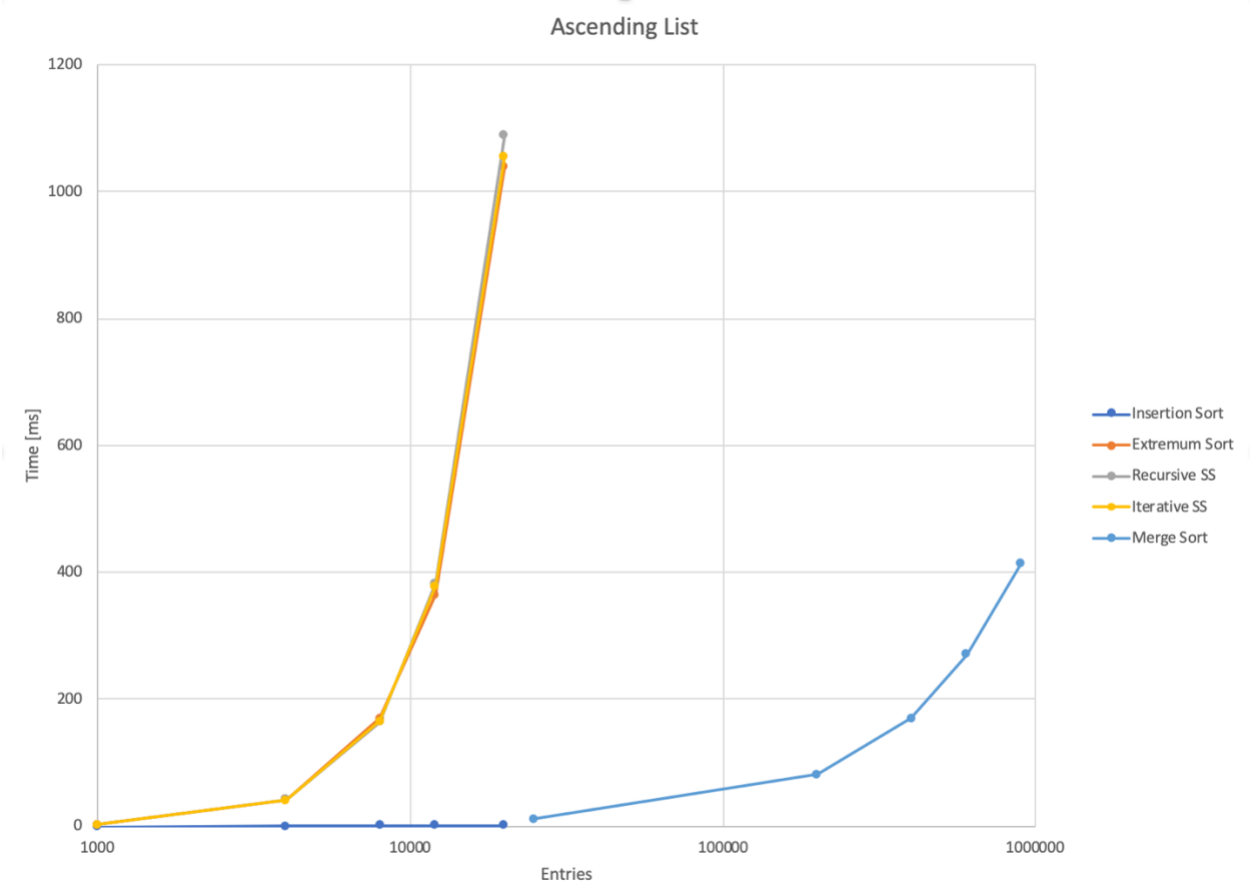
Deeb Armaly
ECE2230

MP3 Test Log

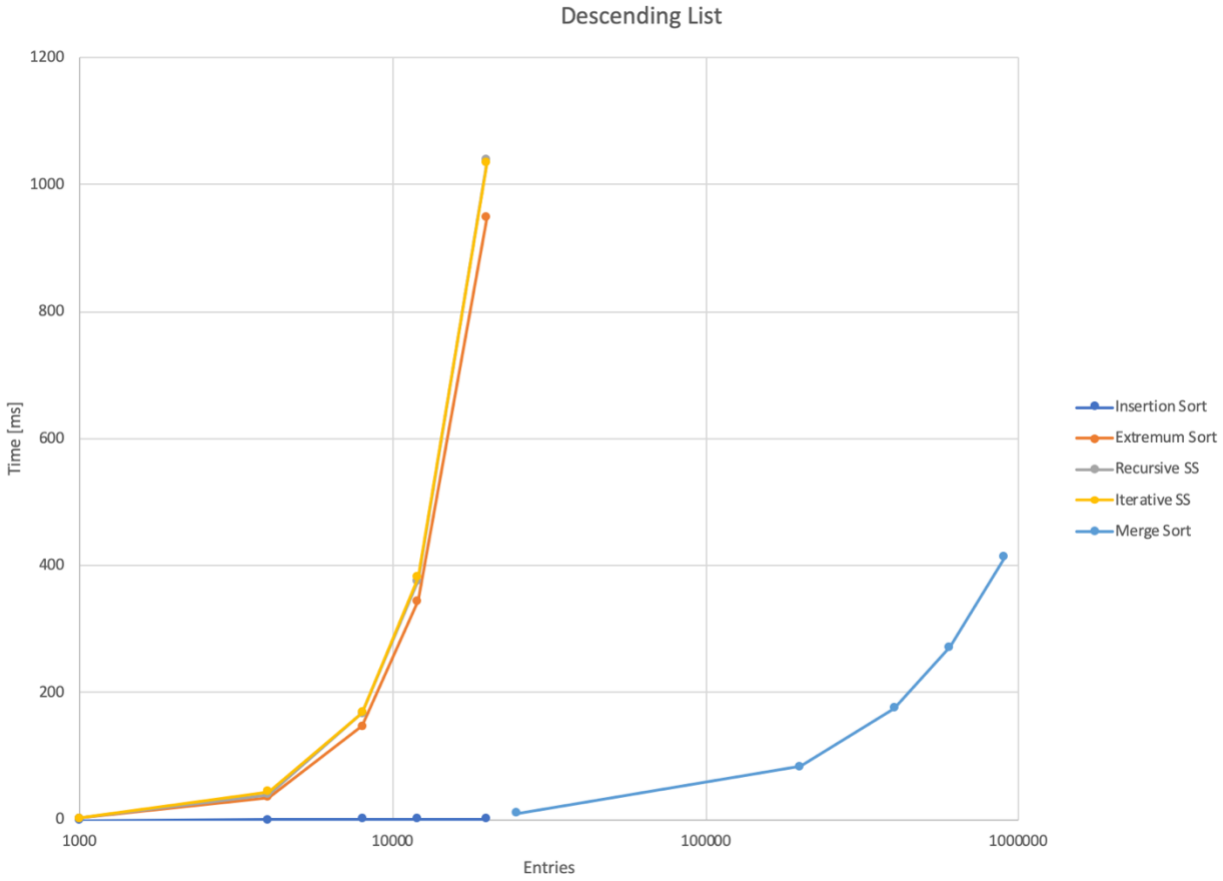
Graph for the different sorting function times with a random list order:



Graph for sorting function times with ascending list order:



Graph for sorting function times with descending list order:



a) For lists that are initially random, the sorting times for all the sorting functions are very similar with the exception of merge sort. Merge sort is by far the quickest for sorting random lists.

b) If the list is already in ascending or descending order, insertion sort proves to be very quick and efficient. Merge sort is also quick at sorting these kinds of lists. Insertion sort is efficient with these lists because it utilizes the linked_insert_sorted function which has quick checks to see if the node is to be inserted at the head or the tail. When the list to be sorted is already in ascending or descending order, the nodes will always have to be inserted at either the head or the tail so insertion sort's utilization of this feature makes it very efficient. Merge sort is very quick because it utilizes a method of splitting the list to be sorted in two with recursion. This leads to the sub lists being broken down to one item very quickly without many function calls on the stack.