Merge Sort Vs. Insertion Sort Analysis

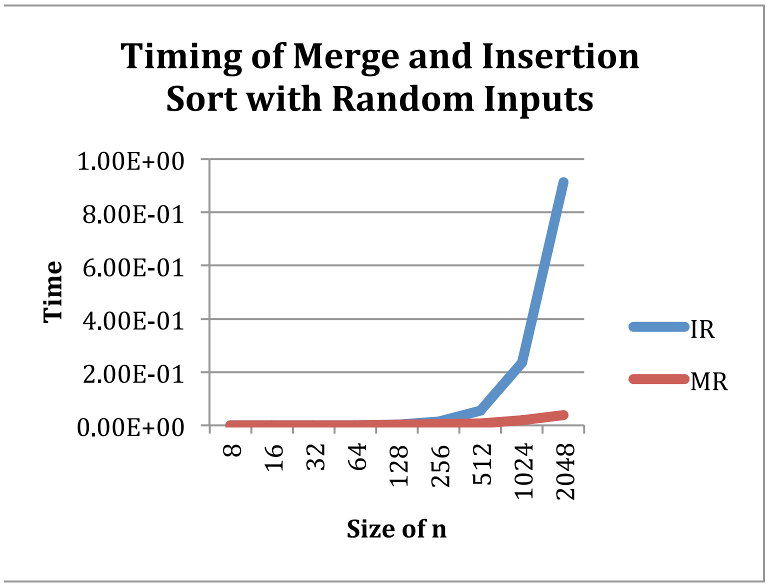
The following conclusions are based on the three graphs below, which are themselves based on the java program that I created to compare running times of the merge sort and insertion sort algorithms. Starting with random inputs, merge sort is clearly faster than insertion sort. As n increases, insertion sort grows at a much faster rate than merge sort does. From studying the algorithm, insertion sort runs at $O(n^2)$ while merge sort runs at O(nlogn).

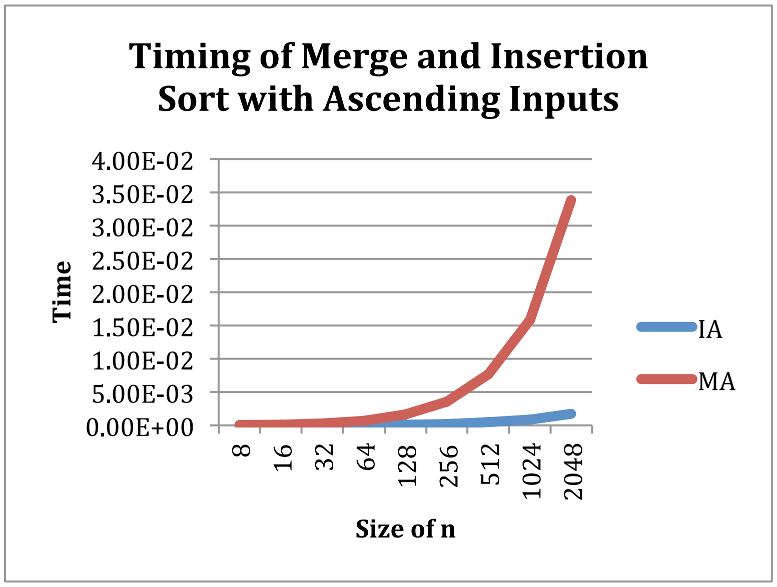
When the values in the array to be sorted are already in ascending order, insertion sort is actually much quicker than merge sort (as shown in the second graph). With large values of n, merge sort increases at a faster rate than insertion sort. This is the best case for insertion sort, which actually runs at O(n) while merge sort runs at O(nlogn) still. Insertion sort only takes O(n) now because for each insertion, there is no shifting of other values required due to it already being sorted.

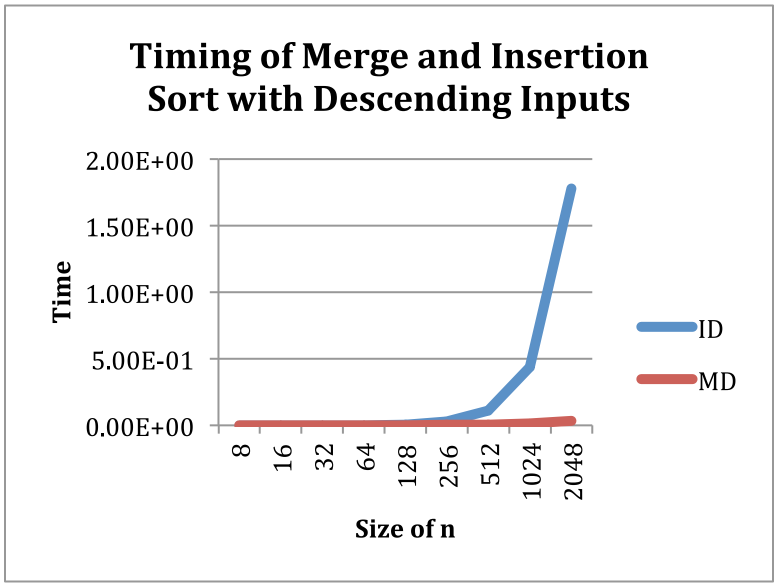
When the values in the array to be sorted are in descending order, merge sort is once again quicker. This is the worst case for insertion sort, because with each insertion you must shift every value already inserted over one by one. Insertion sort is back to $O(n^2)$ and merge sort is still O(nlogn).

Merge sort always has the same time complexity because it is doing the same number of comparisons regardless of how sorted the array is. Insertion sort's best case is when the array is in ascending order, and worst case is when it's in descending order. From analyzing the data, it looks like when the array is random, if n is 32 or less then insertion sort is faster, and if n is 64 or more then merge sort is faster. The actual crossover point is somewhere in between 32 and 64.

Combining the fastest parts of both sorting algorithms would create one that's even faster. A simple way to do this would be to use merge sort up until the array size is 32 or less, and then call insertion sort as a helper method to sort this smaller array. This is opposed to merge sort dividing up the array until each array only has one element.


Timing of Merge and Insertion Sort with Random Inputs
Time
1.00E+00
8.00E-01
6.00E-01
4.00E-01
2.00E-01
0.00E+00
Size of n
IR
MR


Timing of Merge and Insertion Sort with Ascending Inputs
Time
4.00E-02
3.50E-02
3.00E-02
2.50E-02
2.00E-02
1.50E-02
1.00E-02
5.00E-03
0.00E+00
Size of n
IA
MA


Timing of Merge and Insertion Sort with Descending Inputs
Time
2.00E+00
1.50E+00
1.00E+00
5.00E-01
0.00E+00
Size of n
ID
MD