Comparison of comparison sorts

For testing, I created a block which measured run time over 100 loops of the different algorithms to minimize timing imprecision. These testing blocks as well as insertion sort are disabled on my submitted version (in the interest of sorting 5 million elements).

The table below summarizes observed run times on remus for the various sorting algorithms on N random keys (times are in seconds):

N	Insertion × 100	Merge × 100	Quick × 100
500	0.01	0.09	0.07
1000	0.03	0.03	0.01
2500	0.12	0.05	0.04
5000	0.49	0.09	0.08
10000	1.99	0.19	0.12
25000	12.5	0.53	0.41

Merge sort

I designed my merge sort implementation to be from bottom up and non-recursive to minimize the run time stack. Elements are sorted in place and constant space is used.

Crossover with insertion sort

From the run times we see that merge sort is clearly superior to insertion sort for N large, requiring less than 5% of the time required by insertion for N=10000. For N small (¡500), insertion sort wins. The efficiency of these two algorithms are roughly equal for N=1000.

There were minor inconsistencies between repeated experiments, likely due to timing imprecision and randomly generated data for each trial. Both insertion sort and quick sort have different best case and worst case run times so their times depend on the data (which was not constant between repeats). Nevertheless, some general trends are still

Quick sort

Quick sort was implemented to partiton by a median of three for arrays with length ¿ 7 and just use the middle key for those below 7. Instead of copying the partition key to the end, it is copied to the variable pivot and the array is quick sorted by iterating down both ends of the array and swapping elements ¡ pivot with those ¿ pivot on the other side. By iterating down both ends rather than from one end to the other, each element is only swapped a maximum of one time (with exception of pivot itself).

I found quick sort to consistently have runtimes faster than merge sort. However, it occassionally generated times which were inconsistent. This is likely due to quick sort $\in O(n^2)$ while merge sort $\in \Theta(n \log n)$I noticed a general trend where quick sort performed overall consistently better with N large (¿10000) and consistently worse with N small (¡500).