Submit a writeup of your findings (including at least one comparison graph) and code files in a zip file.

**Figure 1.** Comparison of runtime durations for randomly sorted data.

**Figure 2.** Comparison of runtime durations for randomly sorted data (zoomed).

Evidently from figures 1 and 2, the performance of heapsort competes with that of those sorting algorithms determined to be the most time efficient in the project (merge, quick, and shell sort). The empirical behavior of heapsort is approximately linear with a steeper slope than merge, quick, and shell sort. However, the heapsort implementation involves generating a heap from an input array and subsequently sorting the array based on the heap’s root. The heap data structure uses a vector under the hood, hence some of the performance limitations observed can be attributed to the heap generation (the repeated doubling in size of the vector) as opposed to the sorting operation. Heapsorts performance using the described implementation lies somewhere in between that of merge sort and insertion sort.