Assignment 3 Writeup

Michael Coe

October 2021

Abstract

The various sorting algorithms in this assignment were varying in complexity, however, most chalenges in writing the programs followed with fostering a better understanding of syntax and what various issues that can arise in c. While most are armature mistakes, they can cause hours of debugging without a compiler throwing any errors, such as using a single "&" for bolean tests and having uint32_t values work unreliably for indexing arrays compared to int values. On the complexity of the various tests, I found quick sort to be the sweet spot of writing complexity and efficiency out of all of the sorts, as insertion sort was simple yet slow while heap sort was seemingly more complicated than quick sort for a far less efficient method. Shell sort is about as complicated as quick sort, but is only the most efficient at smaller array sizes so had diminishing returns.

Performance

- Insertion Sort
  This was the least efficient sort tested, with exceedingly more compares and moves than any other function. With 100 elements the sort took 2543 moves and 2737 comparisons to complete.

- Heap Sort
  Heap sort bridges the gap between insertion sort and the rest of the sorts, with a noticeable distinction between shell sort and insertion sort, making the sort needlessly inefficient for how many functions required to write for it. With 100 elements the sort took 1755 moves and 1029 comparisons to complete.

- Quick Sort
  This sort lives up to its name by having the smallest number of moves or comparisons out of any sort tested, beating almost every sort by a large margin (shell sort being the exception). With 100 elements the sort only took 1053 moves and 640 comparisons to complete.

- Shell Sort
  Shell sort was the fastest non-recursive sort tested, and matched the efficiency of Quick sort until arrays of about 20 elements. Even at 100 elements quick sort had a very small margin of difference to quick sort taking 1183 mvoes and 801 comparisons to complete

Graphs