Noel Ball
CSE13S

sorting.c Writeup Document

sorting.c:

I built my switch-case statement off of my previous mathlib-test.c test harness. Although it was a bit of a challenge to learn how to use the sets correctly rather than my original implementation of boolean values, I found my errors were in set.h with the creation of the set functions in that I did not originally shift by the correct value. I also struggled with correctly formatting the width of the sorted array values for the prints. The provided pseudo used PRIu32 but I did not know how to format the array's value using it. Reading the PRIu32 man pages, I learned how to use the PRIu32 formatter, and had to test with trial and error to correctly set it to a width of 13.

Makefile:

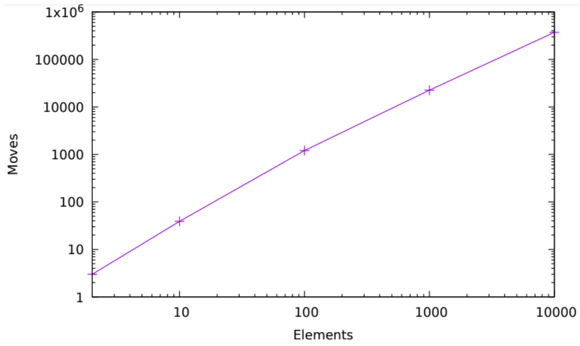
I built my makefile based on mathlib-test's from the previous assignment, and only modified the executable and object files.

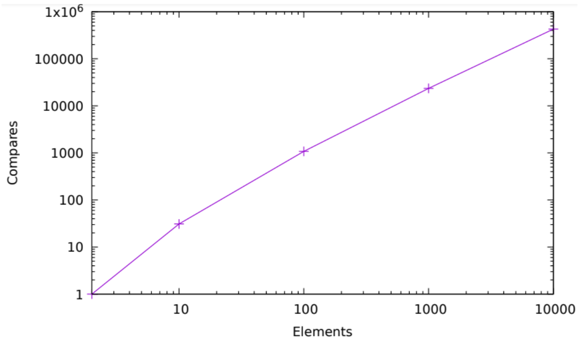
Sorting files;

All my graphs plot the number of elements sorted vs the number of moves or comparisons taken to demonstrate the change in performance from a small to large array scale, ranging from 2 to 10000 elements, plotted with gnuplot on log scale axis.

Noel Ball
CSE13S

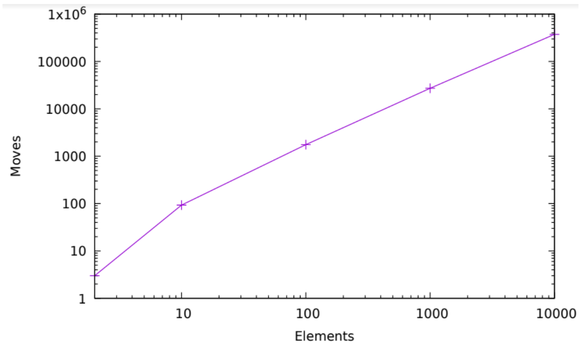
batcher.c:

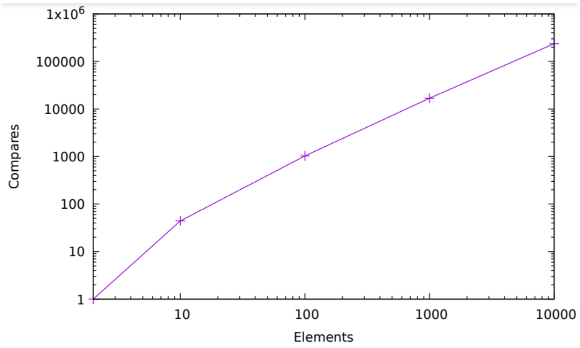




batcher_moves.pdf batcher_compares.pdf

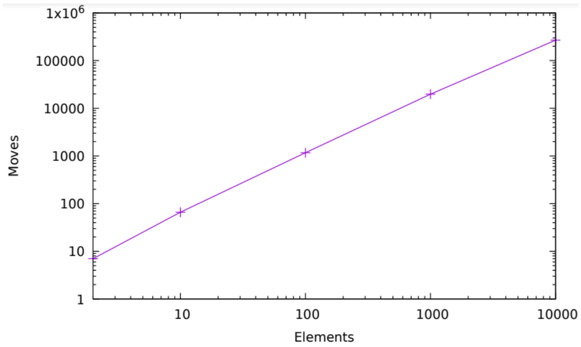
heap.c:

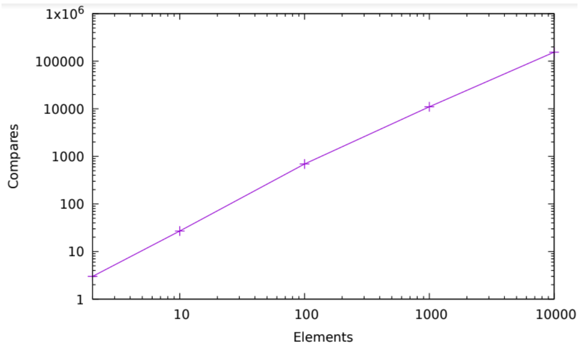




heap_moves.pdf heap_compares.pdf

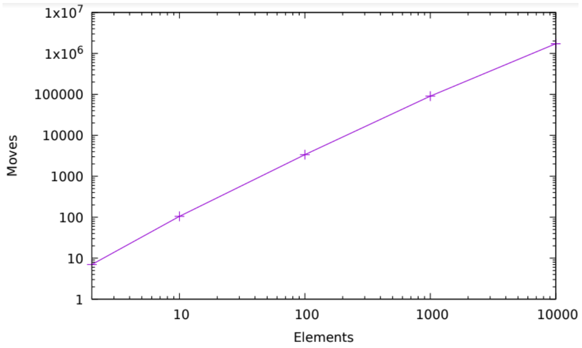
quick.c:

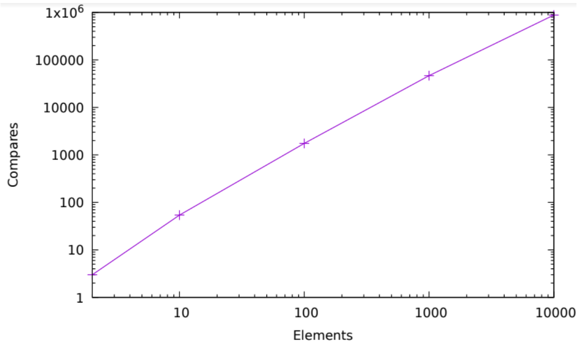




quick_moves.pdf quick_compares.pdf

shell.c:





shell_moves.pdf shell_compares.pdf

Analysis:

Although the graph's log scales make it a little difficult to visualize, shell sort seemed to significantly take more moves and compares than the rest of the sorts with large arrays. Which was surprising to me given that shell sort performed well in the sample data. Shell sort and quicksort both had a high amount of moves and compares with small arrays as well, taking 7 moves versus 3 from the other sorts. I’m not sure what else I can do to optimize my sorting functions but I’m happy with the fact that they all run and correctly sort.