I created a class file for each sorting algorithm. In the case of the ascending.txt and descending.txt files, quicksort was crashing the program when inputs were larger than a certain amount (due to stack overflow).

I couldn’t figure out how to make smoothSort work. I think I am fairly close, but at a hard stop. If you have any ideas please let me know!

Filesize that I used is 200,000. This seemed the most reasonable since inputs of this size already show differences between the different sorting algorithms. Anything above that and the algorithms started being time intensive (about 6-7 hours of runtime for 1M inputs)

If you run the solution, 3 .csv files will be created showing the different runtimes for all three input files. However, I have kept 3 files that will show the data I used to create my graphs. If you want to see the values I used, open “ascending1.csv”, “descending1.csv” and “random1.csv”. Otherwise, once you run the program, the files created will be “ascending.csv”, “descending.csv” and “random.csv”

**See graphs below**. Note: for the more efficient algorithms, I created an additional separate graph to zoom in and better show performance differences.

Ascending.txt

Descending.txt