1 Introduction & Instructions

This lab is another timing analysis, this time between all the array sorts we have recently discussed in class.

When you are finished submit all your work through the MyClasses page for this class. Create a directory called Lab08, put each programming exercise into its own subdirectory of this directory, zip the entire Lab08 directory up into the file Lab08.zip, and then submit this zip file to Lab #8.

Make sure that you include a makefile, check the contents of the zip file before uploading it, and that the file was submitted correctly to MyClasses.

2 Exercise

1. Take the sorting code from the code examples on the MyClasses page and do the following.
   (a) Incorporate timing for merge, quick, comb, shell, heap, radix, count, and bucket sorts. You may use either timer, the chrono library or the one in the `timer.h` file.
   (b) Analyze and report your findings for the average case scenario (random data) for merge, quick, comb, shell, and heap sorts. Experiment with different array sizes, push these as high as possible. We usually just use integer data in the arrays but this time I want you to do two separate experiments one with integer data and one with double data in the range $[0, 1)$.
   (c) Analyze and report your findings for the average case scenario (random data) for the radix sort. Use just positive integer data for these. Compare these results with the integer results from part 1b. Also, allow the user to specify the radix being used and bring that in as a parameter to the radix sort function. Experiment with different array sizes, different radix, and different ranges of data, i.e. data from 1 to n where n is different and input by the user.
   (d) Analyze and report your findings for the average case scenario (random data) for the count sort. Use just positive integer data for these. Compare these results with the integer results from part 1b. Experiment with different array sizes and different ranges of data, i.e. data from 1 to n where n is different and input by the user.
   (e) Analyze and report your findings for the average case scenario (random data) for the bucket sort. Use just double data in the range $[0, 1)$. Compare these results with the double results from part 1b. Also, alow the user to specify the number of buckets being used and bring that in as a parameter to the bucket sort function. Experiment with different array sizes and different numbers of buckets.
2. Create a nice report document (PDF format) summarizing all of your data, graphs, observations, and conclusions. Upload your report with all your code you created to analyze these sorts.