EC 504 – Spring 2023 – Homework 2

Due Wednesday, Feb 22, 11:59PM, 2023. Submit the coding and graphic in the directory `/projectnb/ec504rb/students/yourname/HW2` on your SCC account. The figures will be returned to gradescope. More information in class next week on what to return!

GOAL: This a short exercise to learn how to measure and fit performance to curves. Also to present The result will be presented in graphical form. On the EC504 GitHUB at Plotting_and_Fitting there is information on how to use gnuplot and even in **LittleGnuplot.pdf** instructions on how to install gnuplot in your laptop. This is convenient but all can be done easily using gnuplot on the SCC through the SCC OnDemand interface. Use Slack to exchange helpful hints on graphing!

1 Sorting Problem

The main file on GitHub has a range of sorting algorithms and the ability to construct random lists of varying sizes N. The exercise is to run them for a range of the sizes $N = 16, 32, 64, 128, \cdots$, average over at an ensemble of cases (as much as 100). to get good statistics. Report the average behavior a function of N for each sorting method as to determine the scaling empirically as function of the number the mean number of swap operations vs N.

The main code `sortScaling.cpp` runs the example:

```
insertionSort   mergeSort  quickSort shellSort
```

1.1 Part 2: Sorting Plotting Exercise

The exercise it to make a table of average performance for all 4 algorithm and plot them to see how the scale with N.

For the standard $O(N^2)$ search algorithms involve local (nearest neighbor) exchanges of element of the given list

$$A_{list} = a[0], a[1], a[2], a[3], \cdots, a[N-1] \tag{1}$$

You should find for `insertonSort` you should verify empirically average the algorithm would have

$$\text{Number of Exchanges} \quad = \quad \frac{N(N-1)}{4} \tag{2}$$

For mergeSort it should be exactly $\Theta(NlogN)$ and for quickSort on average $O(NlogN)$. Finally see if you can find the value of the γ for shel sort $O(N^{\gamma})$.

The exercise it to modify the main file to build an out put data file to plot. The code will be tested to see that it does give output but the grade is mostly on the figure and explanation put in gradescope.

```
# N  insertionSort   mergeSort  quickSort shellSort
16   xxxx            xxxx       xxxx      xxxxx
32   xxxx            xxxx       xxxx      xxxxx
64   xxxx            xxxx       xxxx      xxxxx
128  xxxx            xxxx       xxxx      xxxxx
....
```

where `xxxx` are the average values. This is convenient for using gnuplot to plot and fit the curves.

This output file can be make by a `hack` by printing to the standard output. Just run the code in a terminal (aka shell) with `$./sort > datafile.txt` Then you take what you need using an editor. This is useful quick trick, however you should really set up a separate **output** file. This is necessary if you want submit you code in queue. To set up a output file see the example to do this on GitHub at `HW1_codes/makeSortedList.cpp` (Hey basic software technique. Steal method from other codes!)

1.2 Part 2: Get better Statistics and Fit with Error bars

For full credit you should also run statistics to estimate error bars and to a least square fit to estimate the goodness of fit If you have time you could add error bars to the table above and fit with error bars. The average of random cases (called σ) defined as mean square deviation a second Add error bars to the average (called σ) defined as mean square deviation a second column next to the tabulate averages `xxxx`. These are define for each algorithm and size N by

$$\sigma^2 = \frac{1}{N_{trials}-1}\sum_{i=1}^{N_{trials}}(Swaps[i] - AvergeSwap)^2$$

where above we suggested fixing $N_{trials} = 100$. The average numbers of swaps in the 100 trials for each algorithm and size N in the table are:

$$AvergeSwap = \frac{1}{N_{trials}}\sum_{i=1}^{N_{trials}} Swaps[i]$$

You will want to have your code compute the standard error and put into another column in your output file. By the way all these analysis skill will likely come in handy for the team project.

For general background information the `sorting.h` files has a few more sorting algorithms to play with. We could add others like bucket and improve pivots for quicksort etc.

COMMENT: In real time in class there will be help next week on all that is need to do these plots and fits.