CMPS 2200 Recitation 6

In this recitation, we'll look at randomness in computation.

To make grading easier, please place all written solutions directly in `answers.md`, rather than scanning in handwritten work or editing this file.

All coding portions should go in `main.py` as usual.

Determinism versus Randomization in Quicksort

In lecture we saw that adding a random choice of pivot reduced the probability of worst-case behavior for any given input in selection. Let's look at how pivot choices affect Quicksort. For this question, refer to the code in `main.py`

1a)

Complete the implementations of `qsort` and `compare_sort` stubs. Feel free to take from code given in the lectures to help you perform list partitioning. Note that the pivot choice function is input to `qsort`, so you will have to curry `qsort` to test different methods of choosing pivots. Implement variants of `qsort` that correspond to selecting the first element of the input list as the pivot, and to selecting a random pivot. .

.

.

.

1b)

Compare running times using `compare-qsort` between variants of Quicksort and the provided implementation of selection sort (`ssort`). Perform two different comparisons: a comparison between sorting methods using random permutations of the specified sizes, and a comparison using already sorted permutations. How do the running times compare to the asymptotic bounds? How does changing the type of input list change the relative performance of these algorithms? Note that you may have to modify the list sizes based on your system memory; compare at least 10 different list sizes. The `print_results` function in `main.py` gives a table of results, but feel free to use code from Lab 1 to plot the results as well.

enter answers in `answers.md`

1c)

Python uses a sorting algorithm called *Timsort*, designed by Tim Peters. Compare the fastest of your sorting implementations to the Python sorting function `sorted`, conducting the tests in 3b above.

enter answers in `answers.md`