CMPS 101: Winter 2016: Programming HW 1

Due: 22nd January 2016

- The assignment is to be attempted in groups of two.
- Each group needs to submit only one set of solutions.
- Instructions on how to submit the homework will be provided later. For now, set up a git repository and start working on your code.
- The names of the group members, and their UCSC ID (@ucsc.edu email address) should prominently be written in the README file.
- Although no points are given for coding style, unreadable or messy code can be penalized at the grader's option.
- Clearly acknowledge sources, and mention if you discussed the problems with other students or groups. In all cases, the course policy on collaboration applies, and you should refrain from getting direct answers from anybody or any source. If in doubt, please ask the instructors or TAs.

In the class we discussed how the Oh notation provides a pessimistic estimate of the time required by an algorithm to solve a problem in the worst case. Here we will explore this idea on two familiar algorithms, namely insertion sort and merge sort.

- Below you may assume that all the arrays we are dealing with contain integers.
- You need to look up the documentation of the following modules
  - Numpy. In particular Numpy Random.
  - Matplotlib/Pylab for plotting
  - timeit for timing your functions

Question 1 (1 point): Implement the following two functions

- `insertionsort(A)`
- `mergesort(A)`

As you probably already guessed, these functions take as input a `numpy` array A of size n and sort it. Few things to keep in mind:

- Do not use the built in sort function of Python or Numpy.
- Use only elementary operations (e.g., swapping, loops etc).
- Your code should be as close to the pseudo-code we discussed in the class, or as written in the book.

Question 2 (2 point): Create a text file `bestworst.txt` in the repository. For a given size n, explain what the best and worst case input for Insertion sort and Merge sort looks like. Neatly type in your explanation into the text file and submit along with your code.

Question 3 (1 point): For $n \in [1, 2, 3, \ldots, 10,000]$ feed your algorithm the best and worst case input, and store the execution time in an array. Look up the documentation of the `timeit` module in Python to help you with this problem.

Question 4 (1 points): Create two plots, each plot should contain two functions. The first plot shows the execution time of insertion sort with the best and worst inputs for different values of n. The second plot shows the same information, but for merge sort. What do you observe? Save your plots as PDF files with the names `insert.pdf` and `merge.pdf`, and write your comments into a file named `comments.txt`. Submit all three files as part of your homework.

Question 5 (1 points) For $n \in [1, 2, 3, \ldots, 10,000]$ create an array of size n, which contains random integers. Look up the `numpy.random` module for more information on how to do this. Use the `numpy.random.permutation` function to permute your random array. Remember to seed your random number generator with a deterministic number so that your results are reproducible! For each value of n, generate 1,000 random permutations of your array, and give it as input to your sorting code. Find the average time that it takes. Plot this as a function of n. How does the average case behavior of your algorithm differ from the worst and best case behavior? You may add the average case plot to the `insert.pdf` and `merge.pdf` files, and your comments about the average case behavior to the `comments.txt` file.