When working on this quiz, recall the rules stated on the Academic Integrity statement that you signed. You can download the **q8helper** project folder (available for Friday, on the **Weekly Schedule** link) in which to write/test/debug your code. Submit your completed **q8solution** modules (**81**, **82**) and **empirical.pdf** online by Thursday, 11:30pm. I will post my solutions to Ed Discussion reachable via the **Solutions** link on Friday morning.

1a. (5 pts) Write a script that uses the **Performance** class to generate data that we can use to determine empirically the complexity class of the **find_influencers** function defined in the **influence.py** module (my solution to problem 1 of Programming Assignment #1). Call the **evaluate** and **analyze** functions on an appropriately constructed **Performance** class object for **random** graphs constructed from **100** nodes, **200** nodes, … up to **12,800** nodes (on graphs that have **five times the number of edges** as nodes) using a loop to **double the number of nodes** each time. Do **5** timings for each size: each of the 5 timings in **Performance** should run on a **different** random graph, and creating the random graph should not be timed. Hint: Write a script with a **create_random** function that stores a random graph (see the **random_graph** function in the **q81solution.py** module) of the correct size into a **global** name, then time the **find_influencers** function using that global name as an argument. If an exception is raised for any size, print an error message for that size but continue collecting data: this might happen for small sizes, which might be timed to take 0 seconds. I had about **30** lines in my module (including blank lines). See the file **sample8.pdf** (included in the download) for what your output should look like: of course, your times will depend on the speed of your computer (but the complexity class estimation will not). The process can take a few minutes. **Do not time the execution of the random_graph function!**

1b. (3 pts) Fill in part 1b of the **empirical.doc** document (included in the download) with the data that you collect (or use the data in **sample8.pdf** if you cannot get your code to produce the correct results) and draw a conclusion about the complexity class of the **find_influencers** function by seeing how much time it takes to run as the size (in number of nodes) of its input graph doubles. Then predict how long this function will take when running on an input graph of 1 million nodes.

2a. (5 pts) Write a script that uses the **cProfile** module to profile all the functions called when the **find_influencers3** function (a faster version of **find_influencers**) is run on a random graph that has been constructed with **10,000** nodes and **50,000** edges. Generate the random graph first (do not include its generation in the profile information) and then call **CProfile.run** so that it runs **find_influencers3** on that graph; also specify a second argument, which is the file to put the results in (and the file on which to call **pstats.Stats**) to print the results.

For the first one, sort the results decreasing by **ncalls** (print at most the top 20); for the second one, sort the results decreasing by **tottime** (print at most the top 20). Hint: The notes show how to instruct the profiler put the profile information into a file and then show how to access that file and format the results it displays to the console; I had 23 lines in my module (including blank lines). It should take only a few seconds to profile this code. See the file **sample8.pdf** (included in the download) for what your output should look like: of course, your counts and times will depend on the speed of your computer and the random graph generated.

2b. (2 pts) Answer the questions in part 2b of the **empirical.doc** document (included in the download) with the data that you collect (or the data in **sample8.pdf** if you cannot get your code to produce the correct results).

After editing empirical.doc for parts 1b and 2b, convert it into a .pdf document and submit the document in that format on checkmate.