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 on Saturday.

---

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 **spanning_tree** function defined in the **graph_goody** module. Call the **evaluate** and **analyze** functions on an appropriately constructed **Performance** class object for a **random** graph (see **graph_goody**) constructed from **1,000** nodes, **2,000** nodes, … up to **128,000** nodes (and **10** times as many edges as nodes) using a loop to **double the number of nodes** each time. Do **5** random timings for each size. Hint: Write a script with a **create_random** function that stores a random graph (see the **random_graph** function in the **graph_goody** module) of the correct size into a **global** name, then time the **spanning_tree** function using that global name as an argument; I had **17** 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. The process can take a few minutes. Time only the running of the **spanning_tree** function, **not** the creation of the random graph.

1b. (3 pts) Fill in part 1b of the **empirical.doc** document (included in the download; edit/submit this file) 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 **spanning_tree** function by seeing how much time it takes to run as the size (measured by number of nodes) of its input graph doubles.

2a. (5 pts) Write a script that uses the **cProfile** module to profile all the functions called when the **spanning_tree** function is run on a random graph that has been constructed with **15,000** nodes (and 10 times as many edges as nodes for each graph). Generate the random graph, and then call **CProfile.run** so that it runs **spanning_tree** 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. Collect runtime information for only the running of the **spanning_tree** function, **not** the creation of the random graph.

For the first output, print the top 10 results, sorted decreasing by **ncalls**; for the second output, print the top 10 results, sorted decreasing by **tottime**. Hint: The notes show how to instruct the profiler to put the profile information into a file and then show how to access that file and format the results it displays to the console. 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.

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).

**1b and 2b: After editing empirical.doc for parts 1b and 2b, convert it into a .pdf document and submit the document in that format on checkmate. The TAs must be able to read your document on their computers, so don't submit it in any other format than a Word Document or a .pdf file. You can use the lab machines to convert a Word Document to a .pdf file.**