Complex Systems 535/Physics 508: Homework 4

1. What (roughly) is the time complexity of:

   (i) Vacuuming a carpet if the size of the input to the operation is the number n of square feet of carpet?

   (ii) Finding a word in a (paper) dictionary if the size of the input is the number n of words in the dictionary?

   Explain your reasoning in each case.

2. For a network of n vertices show that:

   (i) It takes time $\mathrm{O}(n^2)$ to multiply the adjacency matrix into an arbitrary vector if the network is stored in adjacency matrix form, but only time $\mathrm{O}(m)$ if it is in adjacency list form.

   (ii) It takes time $\mathrm{O}(n(n+m))$ to find the diameter of the network.

   (iii) It takes time $\mathrm{O}(\langle k \rangle)$ to list the neighbors of a vertex, on average, but time $\mathrm{O}(\langle k^2 \rangle)$ to list the second neighbors. You can assume the network is stored in adjacency list format. (In a network with a power-law degree distribution where $\langle k \rangle$ is finite but $\langle k^2 \rangle$ formally diverges, this means the second operation is much more work than the first.)

   (iv) For a directed network in which in- and out-degrees are uncorrelated, it takes time $\mathrm{O}(m^2/n)$ to calculate the reciprocity of the network. Why is the restriction to uncorrelated degrees necessary? What could happen if they were correlated?

3. What is the time complexity, as a function of the number n of vertices and m of edges, of the following network operations if the network in question is stored in adjacency list format?

   (i) Calculating the mean degree.

   (ii) Calculating the median degree.

   (iii) Calculating the air-travel route between two airports that has the shortest total flying time, assuming the flying time of each individual flight is known.

   (iv) Calculating the minimum number of nodes that would have to fail to disconnect two (otherwise functional) nodes on the Internet.

4. Consider the following centrality measure, which I just made up off the top of my head. With ordinary degree centrality, you get points for each person you are connected to. But perhaps you could get points for people you are two steps away from in the network, or three, or more, although probably you should get fewer points the further away someone is. So let us define the centrality x_i of vertex i to be a sum of contributions as follows: 1 for yourself, α for each person at distance 1 in the network, α^2 for each person at distance 2, and so forth.

   (i) Write an expression for x_i in terms of α and the geodesic distances d_{ij} between vertex pairs.

(ii) Describe briefly an algorithm for calculating this centrality measure. What is the time complexity for calculating x_i for all i?

(iii) Suppose individuals in a network have about c connections on average, so that a person typically has about c first neighbors, c^2 second neighbors, and so on (ignoring the effects of transitivity). What happens to the contributions to the centrality when $\alpha \gtrsim 1/c$?

5. **Extra credit programming challenge:** This question is optional. You can get 100% on this week's homework without doing it (but you'll get extra credit if you do it).

   The file `http://www.umich.edu/~mejn/courses/2012/cscs535/internet.txt` contains a snapshot of the structure of the Internet at the autonomous system level in adjacency list format. The $n = 22\,963$ vertices in the network are numbered in order from zero (note: not from 1) up to 22962, with one line of the file for each one. A single line of the file looks, for example, like this:

   ```
   112 3 12 22 24
   ```

   This means that vertex 112 has degree 3, and its three neighbors are the vertices numbered 12, 22, and 24.

   Write a program in the programming language of your choice to do the following:

   (i) Read the data from the file into a data structure of your choice, to represent the network, in adjacency list format, in the memory of the computer. For example, in C I might use a dynamically allocated 2D integer array with a line for each vertex, and another 1D array for the degrees. In Python I might use an array of n lists or sets to store the neighbors of each vertex, and an array of integers for the degrees.

   (ii) Calculate, using breadth-first search, the shortest distance from a given single vertex to every other, then average those distances to calculate the closeness centrality of the given vertex.

   Use your program to calculate the closeness centrality of the vertex numbered 0 in the network.

   For full extra credit, turn in a complete printout of your program, and a printout of it in action, showing the answer it finds.

   (Note: There are programs or libraries that you could download from the Internet that will do the closeness calculation for you. While it is perfectly legitimate under other circumstances to make use of such programs, doing so will not get you any credit on this question. You have to actually write your own breadth-first search program to get credit on this question.)