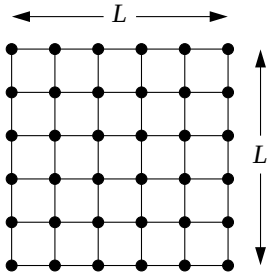
Complex Systems 535/Physics 508: Homework 2

1. One can calculate the diameter of certain types of networks exactly:

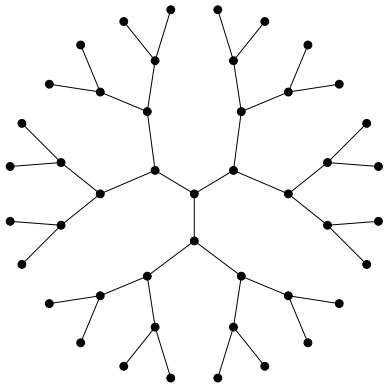
   (i) What is the diameter of a clique?

   (ii) What is the diameter of a square portion of square lattice, with L edges (or equivalently $L+1$ vertices) along each side, like this:



   What is the diameter of the corresponding hypercubic lattice in d dimensions with L edges along each side? Hence what is the diameter of such a lattice as a function of the number n of vertices?

   (iii) A Cayley tree is a symmetric regular tree in which each vertex is connected to the same number k of others. Here, for example, is a Cayley tree for $k = 3$:



   Show that the number of vertices reachable in d steps from the central vertex is $k(k-1)^{d-1}$ for $d \geq 1$. Hence find an expression for the diameter of the network in terms of k and the number of vertices n.

   (iv) Which of the networks in parts (i), (ii), and (iii) displays the small-world effect, defined as having a diameter that increases as $\log n$ or slower?

2. A particular network is believed to have a degree distribution that follows a power law for degree greater than or equal to 10. A random sample of vertices is taken and their degrees measured. The degrees of the first twenty vertices with degrees 10 or greater are:

16	17	10	26	13
14	28	45	10	12
12	10	136	16	25
36	12	14	22	10

Estimate the exponent α of the power law and the error on that estimate.

3. 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.

4. For a network of n vertices show that:

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

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

   (iii) It takes time $O(\langle k \rangle)$ to list the neighbors of a vertex, on average, but time $O(\langle k^2 \rangle)$ 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 $O(m^2/n)$ to calculate the reciprocity of the network. Why is the restriction to uncorrelated degrees necessary? What could happen if they were correlated?

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/2013/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 you might use a dynamically allocated 2D integer array with a line for each vertex, and another 1D array for the degrees. In Python you 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) Write code to 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.)