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 $O(n^2)$ 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?

3. (If you have not already done so, you should read the section on "Heaps" in the course-pack, Section 9.7.)

   An active line of research in the late 90s concerned what would happen to a network if you disabled or attacked its vertices one by one, a question of relevance for instance to the vaccination of populations against the spread of disease. A typical approach would be to remove vertices in order of their degrees, starting with the highest degrees first. Note that once you remove one vertex (along with its associated edges) the degrees of some of the other vertices may change.

   (i) What is the time complexity of finding the highest-degree vertex in a network, assuming the vertices are given to you in no particular order?

   (ii) If we perform the repeated vertex removal in a dumb way, searching exhaustively for the highest-degree vertex, removing it, then searching for the next highest, and so forth, what is the time complexity of the entire operation?

   (iii) Suppose you store the degrees of the vertices in a heap. What then is the time complexity of the entire operation?

   (iv) More generally, but taking the same approach, describe in a sentence or two a method for taking n numbers in random order and sorting them into decreasing order using a heap. Show that the time complexity of this sorting algorithm is $O(n \log n)$.

   (v) The degrees of the vertices in a simple graph are integers between zero and n. It is possible to sort such a set of integers into numerical order, either increasing or decreasing, in time $O(n)$. Describe briefly an algorithm that achieves this feat.

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 $\alpha \gtrsim 1/c$?