Problem 1. Consider a growing random network model generated as follows:

- At each time step, k nodes ($k > 1$) each with degree k are born.
- All k of the newborn nodes connect to each other (i.e., the subgraph of these k nodes is a complete graph).
- Each newborn node connects its last link to the existing graph, according to the rules of the preferential attachment model.

What is the degree distribution in the graph obtained by this process? How does the resulting graph compare to the one obtained by the preferential attachment process? (assume reasonable starting configuration)

Problem 2. Consider a one-dimensional analog of Kleinberg's grid-based navigation model. Nodes are ordered on a line and each node i has a single long-range (directed) edge, which links it with a node j, with probability proportional to $1/|i-j|^{\alpha}$. Proceed as in the lecture to determine the navigation times for different values of the clustering exponent α.

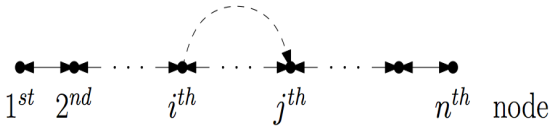

Figure 1: 1-D grid model.

Problem 3. *(Problem 7.2 from Jackson)*
Consider the SIR model in a situation in which a proportion π of the nodes (selected uniformly at random) are immune from the outset. Develop an estimate of the threshold for infection, relating π and the probability of transmission t to statistics of the degree distribution. Plot combinations of π and t at the threshold for several values of average degree in the Erdös-Renyi random graph model.

Problem 4. *(Small world network vs. Random graph: Computational Problem)*
The goal of this computational problem is to introduce you to Pajek, a social network analysis tool. You can download Pajek from http://vlado.fmf.uni-lj.si/pub/networks/pajek/. A tutorial for Pajek, prepared by Prof. Lada Adamic from the University of Michigan, can be found at the Stellar website.
In the problem, we will study some properties of small world networks and contrast them with random graphs. Here is the description:

(1) Download the file lattice.net from the Stellar website.

(2) Lay out the network using any of the pre-built algorithms (e.g., Kamada-Kawai).

(3) Compute the clustering coefficient and average shortest path.

(4) Then, add 10 random edges, one by one, each time recording the clustering coefficient and average shortest path - plot and turn in the ratio of each value relative to that of the lattice graph (in case you need a random number generator you can go to http://www.random.org/nform.html. Then either edit the .net file to add the edges, or right click on the randomly chosen node in the visualization, then click on 'newline' and type in the number of the other random node it is supposed to connect to.)

(5) What percentage of the edges are now random? Submit an image of the resulting network.

(6) Create a Erdos-Renyi graph in Pajek with the same number of nodes and average degree as the lattice graph. Compute the clustering coefficient and average shortest path. How close did the lattice graph with the 10 random edges come to having the same average clustering as the random graph? How about the average shortest path?