Project 11.2 Slowing an epidemic

In Section 4.4, we simulated the spread of a flu-like virus through a population using the SIR model. In that simulation, we lumped all individuals into three groups: those who are susceptible to the infection, those who are currently infected, and those who recovered. In reality, of course, not every individual in the susceptible group is equally susceptible and not every infected person is equally likely to infect others. Some of the differences have to do with who comes into contact with whom, which can be modeled by a network. For example, consider the network in Figure 11.6 on Page 458. In that network, if node 3 becomes infected and there is a 10% chance of any individual infecting a contact, then the virus is very unlikely to spread at all. On the other hand, if node 12 becomes infected, there is a $9 \cdot 10\% = 90\%$ chance that the virus will spread to at least one of the node's nine neighbors.

If a vaccine is available for the virus, then knowledge about the network of potential contacts can be extremely useful. In this project, you will simulate and compare two different vaccination strategies: vaccinating randomly and vaccinating targeted nodes based on a criterion of your choosing.

Part 1: Create the network

For this project, you will use a network that is an anonymized version of a very small section of the Facebook network with 333 nodes and 2,519 links. The file containing this network is available on the book website. The format of the file is the same as the format described in Exercise 11.1.7: each node is represented by a number, and each line represents one link. If you have not already done so in Exercise 11.1.7, write a function

```
readNetwork(fileName)
```

that reads in a network file with this format, and returns an adjacency list representation of the network.

Part 2: Simulate spread of the contagion through the network

Simulating the spread of the infection through the network is very similar to a breadth-first search, except that nodes are visited (infected) probabilistically and we will assume that each infection starts with a random source node. Once a node has been infected, it should be enqueued so that it has an opportunity to infect other nodes. When a node is dequeued, it should attempt to infect all of its uninfected neighbors, successfully doing so with some *infection probability* p. But once infected, a node should never be infected again or put back into the queue. Using the `bfs` function from Section 11.2 as a starting point, write a function

```
infection(network, p, vaccinated)
```

that simulates this infection spreading in `network`, starting from a randomly chosen source node, and using infection probability `p`. The fourth parameter, `vaccinated`, is a list of nodes that are immune to the infection. An immune node should never become infected, and therefore cannot infect others either; more on this in Part 3.

The function should return when all infected nodes have had a chance to infect their neighbors. The function should return the total number of nodes who were infected.

Part 3: Simulate vaccinations

Now, you will investigate how the number of infected nodes is affected by vaccinations. Assume that the number of vaccine doses is limited, and compare two different strategies for choosing the sequence of nodes to vaccinate. For each strategy, first run your simulation with no vaccinations, then with only the first node vaccinated, then with the first two nodes vaccinated, etc. Because the infection is a random process, each simulation with a different number of vaccinations needs to be repeated many times to find an average number of infected nodes.

First, investigate what happens if you vaccinate a sequence of random nodes. Second, design a better strategy, and compare your results to the results of the random vaccinations. Use the ideas from this chapter to learn about the network, and choose a specific sequence of nodes that you think will do a better job of reducing the number of infections in the network.

Implement these simulations in a function

```
vaccinationSim(network, p, trials, numVaccs, randomSim)
```

If the last parameter `randomSim` is `True`, the function should perform the simulation with random vaccinations. Otherwise, it should perform the simulation with your improved strategy. The fourth parameter, `numVaccs`, is the number of individuals to vaccinate. For every number of vaccinations, the simulation should call the `infection` function `trials` times with the given `network` and infection probability `p`. The function should return a list of `numVaccs + 1` values representing the average number of infections that resulted with $0, 1, 2, \ldots,$ `numVaccs` immune nodes.

Create a program that calls this function once for each strategy with the following arguments:

- infection probability 0.1
- 500 trials
- 50 vaccinations

Then your program should produce a plot showing for both strategies how the number of infections changes as the number of vaccinations increases.

Question 11.2.1 *What strategy did you choose?*

Question 11.2.2 *Why did you think that your strategy would be effective?*

Question 11.2.3 *How successful was your strategy in reducing infections, compared to the random strategy?*

Question 11.2.4 *Given an unlimited number of vaccinations, how many are required for the infection to not spread at all?*