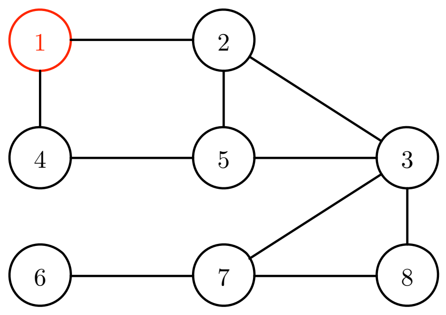
11.6 PROJECTS

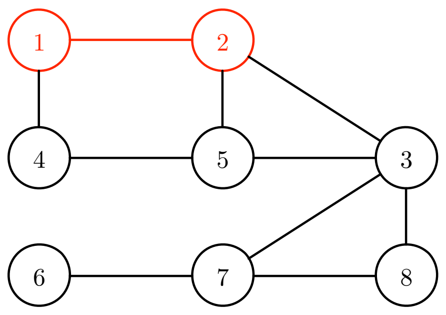
Project 11.1 Diffusion of ideas and influence

In this project, you will investigate how memes, ideas, beliefs, and information propagate through social networks, and who in a network has the greatest influence. We will simulate diffusion through a network with the ***independent cascade model***. In this simplified model, an idea originates at one node (called the seed) and each of the neighbors of this node adopts it with some fixed probability (called the *propagation probability*). This probability measures how influential each node is. In reality, of course, people have different degrees of influence, but we will assume here that everyone is equally influential. Once the idea has propagated to one or more new nodes, each of these nodes gets an opportunity to propagate it to each of their neighbors in the next round. This process continues through successive rounds until all of the nodes that have adopted the idea have had a chance to propagate it to their neighbors.

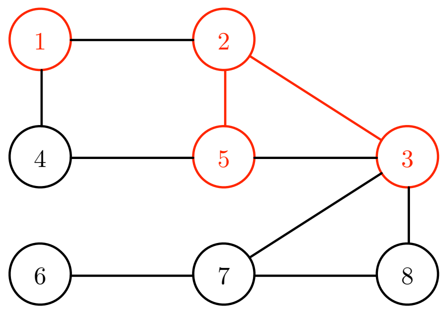
For example, suppose node 1 in the network below is the seed of an idea.



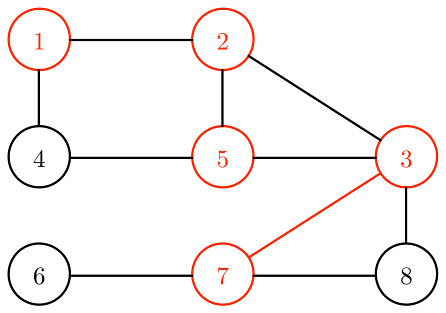
Initial state



Round 1



Round 2



Round 3

In round 1, node 1 is successful in spreading the idea to node 2, but not to node 4. In round 2, the idea spreads from node 2 to both node 3 and node 5. In round 3, the idea spreads from node 3 to node 7, but node 3 does not successfully influence node 8. Node 5 also attempts to influence node 4, but is unsuccessful. In round 4 (not shown), node 7 attempts to spread the idea to nodes 6 and 8, but is unsuccessful, and the process completes.

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 diffusion with a single seed

Simulating the diffusion of an idea through the network, as described above, is very similar to a breadth-first search, except that nodes are visited probabilistically. Just as visited nodes are not revisited in a BFS, nodes that have already adopted the idea are not re-influenced. Using the `bfs` function from Section 11.2 as a starting point, write a function

```
diffusion(network, seed, p)
```

that simulates the independent cascade model starting from the given seed, using propagation probability `p`. The function should return the total number of nodes in the network who adopted the idea (i.e., were successfully influenced).

Part 3: Rate each node's influence

Because this is a random process, the `diffusion` function will return a different number every time it is run. To get a good estimate of how influential a node is, you will have to run the function many times and average the results. Therefore, write a function

```
rateSeed(network, seed, p, trials)
```

that calls `diffusion` `trials` times with the given values of `network`, `seed` and `p`, and returns the average number of nodes influenced over that many trials.

Part 4: Find the most influential node(s)

Finally, write a function

```
maxInfluence(network, p, trials)
```

that calls the `rateSeed` function for every node in the network, and returns the most influential node.

Combine these functions into a complete program that finds the most influential node(s) in the small Facebook network, using a propagation probability of 0.05. Then answer the following questions. You may write additional functions if you wish.

Question 11.1.1 *Which node(s) turned out to be the most influential in your experiments?*

Question 11.1.2 *Were the ten most influential nodes the same as the ten nodes with the most friends?*

Question 11.1.3 *Can you find any relationship between a person's number of friends and how influential the person is?*

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?*

Project 11.3 The Oracle of Bacon

In the 1990's, a group of college students invented a pop culture game based on the six degrees of separation idea called the "Six Degrees of Kevin Bacon." In the game, players take turns challenging each other to identify a connected chain of actors between Kevin Bacon and another actor, where two actors are considered to be connected if they appeared in the same movie. An actor's Bacon number is the distance of the actor from Kevin Bacon.

This game later spawned the "Oracle of Bacon" website, at `http://oracleofbacon.org`, which gives the shortest chain between Mr. Bacon and any other actor you enter. This works by constructing a network of actors from the Internet Movie Database (IMDb)[7], and then using breadth-first search to find the desired path. This network of movie actors provides fertile ground for investigating all of the concepts discussed in this chapter. Is the network really a small-world network, as suggested by the "Six Degrees of Kevin Bacon?" Is there really anything special about Kevin Bacon in this network? Or is there a short path between any two pair of actors? Is the network scale-free?

In this project, you will create an actor network, based on data from the IMDb, and then answer questions like these. You will also create your own "Oracle of Bacon."

Part 1: Create the network

On the book website, you will find several files, created from IMDb database files, that have the following format:

```
League of Their Own, A (1992)▷Tom Hanks▷Madonna▷Penny Marshall▷ ···
Animal House (1978)▷Kevin Bacon▷John Belushi▷Donald Sutherland▷ ···
Apollo 13 (1995)▷Kevin Bacon▷Tom Hanks▷Bill Paxton▷Gary Sinise▷ ···
```

The files are tab-separated (the ▷ symbols represent tabs). The first entry on each line is the name of a movie. The movie is followed by a list of actors that appeared in that movie.

Write a function

```
createNetwork(filename)
```

that takes the name of one of these data files as a parameter, and returns an actor network (as an adjacency list) in which two actors are connected if they have appeared in the same movie. For example, for the very short file above, the network would look like that in Figure 1. Each link in this network is labeled with the name of a movie in which the two actors appeared. This is not necessary to determine a Bacon number, but it is necessary to display all the relationships, as required by the traditional "Six Degrees of Kevin Bacon" game (more on this later).

The following files were created from the IMDb data, and are available on the book website.

[7] `http://www.imdb.com`

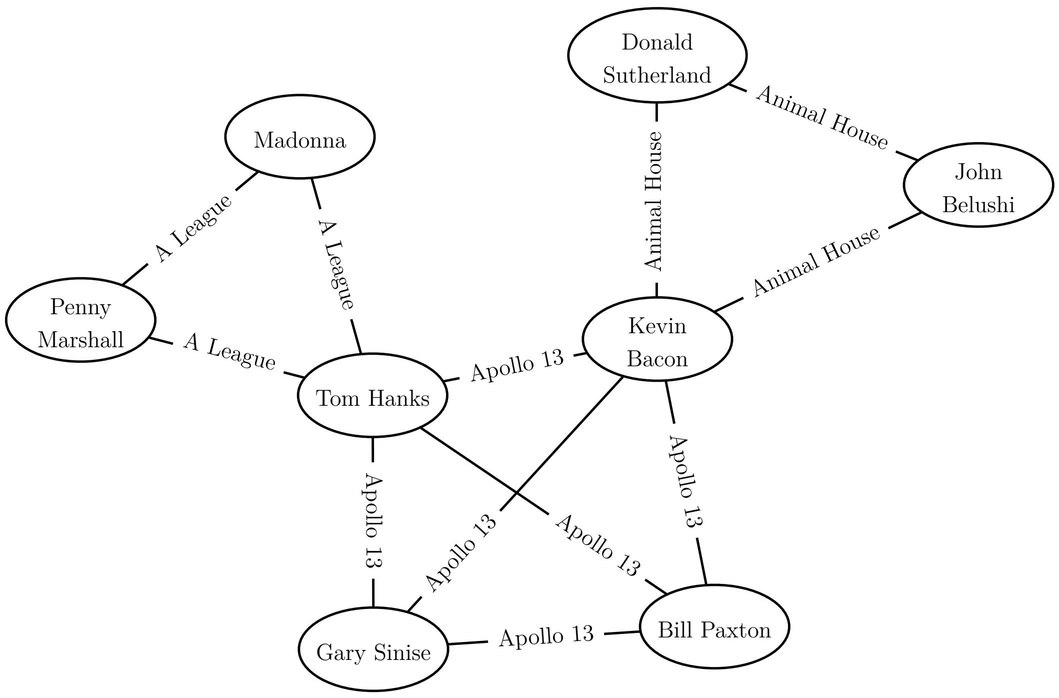


Figure 1 A simple actors network.

File	Movies	Actors	Description
`movies2005.txt`	560	24,276	MPAA-rated movies from 2005
`movies2012.txt`	518	25,225	MPAA-rated movies from 2012
`movies2013.txt`	517	24,190	MPAA-rated movies from 2013
`movies2000p.txt`	8,396	362,798	MPAA-rated movies from 2000–2014
`movies2005p.txt`	5,590	251,881	MPAA-rated movies from 2005–2014
`movies2010p.txt`	2,547	117,908	MPAA-rated movies from 2010–2014
`movies_mpaa.txt`	12,404	514,780	All movies rated by MPAA
`moves_all.txt`	565,509	6,147,773	All movies

Start by testing your function with the smaller files, building up to working with the entire database. Keep in mind that, when working with large networks, this and your other functions below may take a few minutes to execute. For each file, we give the number of movies, the total number of actors in all casts (not unique actors), and a description.

Part 2: Is the network scale-free?

Because the actor network is so large, it will take too much time to compute the average distance between its nodes and its clustering coefficient. But we can plot the degree distribution and investigate whether the network is scale-free. To do so, write a function

```
degreeDistribution(network)
```

that plots the degree distribution of a network using `matplotlib`. Again, start with small files and work your way up.

Question 11.3.1 *What does your plot show? Is the network scale-free?*

Question 11.3.2 *Where does Kevin Bacon fall on your plot? Is he a hub? If so, is he unique in being a hub?*

Question 11.3.3 *Which actors have the ten largest degrees? (They might not be who you expect.)*

Part 3: Oracle of Bacon

You now know everything you need to create your own "Oracle of Bacon." Write a function

```
oracle(network)
```

that repeatedly prompts for the names of two actors, and then prints the distance between them in the network. Your function should call `bfs` to find the shortest distance.

For an extra challenge, modify the algorithm (and your adjacency list) so that it also prints how the two actors are related. For example, querying the oracle with Kevin Bacon and Zooey Deschanel might print

```
Zooey Deschanel and Kevin Bacon have distance 2.

1. Kevin Bacon was in "The Air I Breathe (2007)" with Clark Gregg.
2. Clark Gregg was in "(500) Days of Summer (2009)" with Zooey Deschanel.
```

Part 4: Frequencies of Bacon numbers

Finally, determine just how central Kevin Bacon is in the movie universe. To do so, create a chart with the frequency of Bacon numbers. Write a function

```
baconNumbers(network)
```

that displays this chart for the given network. For example, given the file `movies2013.txt`, your function should print the following:

```
Bacon Number   Frequency
------------   ---------
     0                 1
     1               152
     2              2816
     3             12243
     4              4230
     5               253
     6                47
     7                 0
     8                 0
  infinity           819
```

An actor has infinite Bacon number if he or she cannot be reached from Kevin Bacon.

Once your function is working, call it with some of the larger files.

Question 11.3.4 *What does the chart tell you about Kevin Bacon's place in the movie universe?*