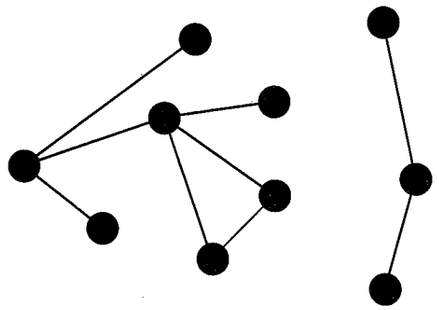

Figure 14.3 Example of a disconnected network with 10 nodes and 9 links. The degree distribution for this network is $D(1) = 5/10 = 0.5$, $D(2) = 3/10 = 0.3$, $D(3) = 1/10 = 0.1$, and $D(4) = 1/10 = 0.1$. The cluster coefficient or transitivity is defined as 3 times the number of triangles divided by the number of possible triples of connected nodes. In this case we have 1 triangle and 12 triples. Thus, the clustering coefficient equals $3 \times 1/12 = 0.25$. If a node has ℓ links, then the number of triples centered at that node is $\ell/(2!(\ell-2)!)$.

spanning cluster as can exist in percolation models. Instead, there can be a cluster that is significantly larger than the other clusters. In the Erdös–Rényi model, the transition at which such a "giant" cluster appears depends on the probability p that any pair of nodes is connected. In the large N limit, this transition occurs at $p = 1/N$.

Problem 14.15 The Erdös–Rényi model

(a) Write a program to create networks based on the Erdös–Rényi model. Choose $N = 100$ and $p \approx 0.01$, and compute $D(\ell)$; average over at least 10 networks. Show that $D(\ell)$ follows a Poisson distribution.

(b) Define a giant cluster as one that has over three times as many nodes as any other cluster and at least 10% of the nodes. Find the value of p at which the giant cluster first appears for $N = 64$, 128, and 256. Average over 10 networks for each value of N. The cluster distribution should be updated after every link is added using the labeling procedure used in Chapter 12. In this case it is easier because every time we add a link, we either combine two clusters or we make no change in the cluster distribution. ■

Some of the networks that we will consider are by definition connected. In these cases one of the important quantities of interest is the mean path length between two nodes, where the path length between two nodes is the shortest number of links from one node to the other. If the mean path length weakly depends on the total number of nodes and is small, then this property of networks is known as the "small world" property. A well-known example of the small world property is what is called "six degrees of separation," which refers to the fact that almost any person is connected through a sequence of six connections to almost any other person.

We wish to understand the structure of different networks. One structural property is the clustering coefficient or transitivity. If node A is linked to B and B to C, the clustering coefficient is the probability that A is linked to C (see Figure 14.3 for a precise definition). If this coefficient is large, then there will be many small loops of nodes in the network. If

we think of the nodes as people and the links as friendship connections, then the clustering coefficient is a measure of the tendency of people to form cliques. It is also of interest to see to what extent the network is hierarchically organized. Can we find groups of nodes that are linked together at different levels of organization? Can we produce an organizational chart for the network similar to what is used by many businesses? Algorithms for computing the hierarchical or community structure of a network are discussed in the references.

Two popular network models are the Watts–Strogatz small world model and the Barabasi–Albert preferential attachment model. In the Watts–Strogatz model, a regular lattice of nodes connected by nearest neighbor links is "rewired" so that a link between two neighboring nodes is broken with probability p, and a link is randomly added between one of the nodes and any other node in the system. The small world property shows up as a logarithmic dependence of the mean path length on the system size N for large p. The degree distribution is similar to that of the Erdös–Rényi model.

In the preferential attachment model, we begin with a few connected nodes and then add one node at a time. Each new node is then linked to m existing nodes, with preference given to those nodes that already have many links. The probability of a node with ℓ links being connected to a new node is proportional to ℓ. For example, if we have ten nodes in the network with 1, 1, 3, 2, 7, 3, 4, 7, 10, and 2 links, respectively, then there are a total of 40 links, and the probability of getting the next link from a new node is 1/40, 1/40, 3/40, 2/40, 4/40, 3/40, 4/40, 7/40, 10/40, and 2/40, respectively. The result of this growth rule is that some nodes will accumulate many links. The key result is that the link distribution is a power law with $D(\ell) \sim \ell^{-\alpha}$. This *scale-free* behavior is very important because it says that in the limit of an infinite network, there is a nonnegligible probability that a node exists with any particular number of links. Examples of real networks that have this behavior are actor networks where the links correspond to two actors appearing in the same movie, airport networks, the internet, and the links between various websites. In addition to the scale-free degree distribution, the preferential attachment model also has the small world property that the mean path length grows only logarithmically with the number of nodes.

The `PreferentialAttachment` class implements the preferential attachment model. Method `setPosition` is not relevant to the actual growth model. It places the nodes in random positions so that they are not too close to each other. This drawing method is useful only for networks less than about 100 nodes.

Listing 14.9 Preferential attachment network model.

```
package org.opensourcephysics.sip.ch14.networks;
import java.awt.*;
import org.opensourcephysics.frames.*;
import org.opensourcephysics.display.*;

public class PreferentialAttachment implements Drawable {
   int[] node, linkFrom, degree;
   // positions of nodes; only meaningful for display purposes
   double[] x, y;
   int N;                          // maximum number of nodes
   int m = 2;                      // number of attempted links per node
   int linkNumber = 0;             // twice current number of links
   int n = 0;                      // current number of nodes
   boolean drawPositions = true; // only draw network if true
   int numberOfCompletedNetworks = 0;
```