2 Small Worlds

path: (*n.*) a sequence of arcs in a network that can be traced continuously without retracing any arc.

Many types of networks display a few fundamental features. In this chapter we introduce three of these characteristics: similarity between neighbors, short paths connecting nodes, and triangles formed by common neighbors. *Social networks* provide us with familiar cases to illustrate these features. In a social network, nodes represent people and links represent some type of social relationship, such as friendship, work, acquaintances, or family ties. Social networks are the most extensively studied category of networks; there is a century of literature on this topic.

2.1 Birds of a Feather

In a social network, nodes may have many properties, such as age, gender identity, ethnicity, sexual preference, location, topics of interests, and so on. Often, nodes that are connected to each other in a social network tend to be similar in their features: for example, relatives may live near each other, and friends may have similar interests. The technical name of this property is *assortativity*. Figure 2.1 illustrates assortativity based on a node feature represented by color. A more striking, real-world example from Twitter is shown in Figure 0.3. Because of assortativity, we are able to make predictions about a person's qualities by inspecting their neighbors. For instance, as we have seen in Section 0.1, researchers found that it is possible to ascertain with reasonable accuracy a Facebook user's sexual orientation and a Twitter user's political preference, even when these features are not present in their profile, by analyzing their circles of friends.

Multiple factors may be responsible for the presence of assortativity in social networks. One possibility is that if people are similar in some way, they are more likely to *select* each other and become connected. This property is captured by a popular proverb: "birds of a feather flock together." Its technical name is *homophily*. Examples include people living in geographic proximity, or who practice the same

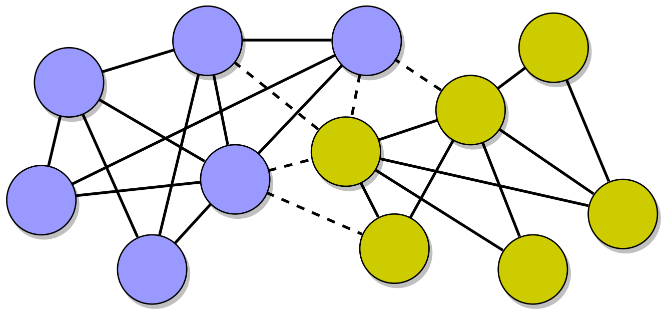


Fig. 2.1 Illustration of network assortativity. Nodes are more likely to be linked to other nodes of the same color than to nodes of different color. In particular, the majority of links for each node go to nodes of the same color, and the majority of links connect nodes of the same color. The few links connecting nodes of different colors are shown as dashed lines.

sport or hobby — these individuals are likely to meet and become friends. Dating apps leverage this kind of homophily by recommending matches based on shared personality traits. The converse mechanism is when people who are friends become more similar over time, through the process of *social influence*. Humans are social animals who tend to imitate each other since birth. Our ideas, opinions, and preferences are strongly affected by our social interactions. It is difficult to separate the causes of assortativity — does similarity induce links, or do links induce similarity? Often these factors simultaneously shape our social connections and reinforce each other. We will return to this question in Chapter 7.

There is a dark side of homophily, too. On social media, it is exceedingly easy to connect with people who share our worldviews and unfriend or unfollow people with different opinions — all it takes is a tap of our finger. Furthermore, information can be shared and consumed in such a selective and efficient way as to influence our opinions very effectively. These mechanisms can lead to the segregation and polarization of our online communities, as is evident from Figure 0.3. When we are surrounded by people who reflect our own views, we are in an *echo chamber* — all the information and opinions to which we are exposed reflect our own and confirm or reinforce our ideas, rather than challenging them. This can be dangerous because it makes us vulnerable to manipulation by misinformation and social bots, as we will see in Chapter 4.

In NetworkX, we can calculate the assortativity of a network based on a given node attribute. There are two functions for cases when the attribute is categorical, such as gender, or numeric, such as age:

```
assort_a = nx.attribute_assortativity_coefficient(G, category)
assort_n = nx.numeric_assortativity_coefficient(G, quantity)
```

Assortativity is not exclusive of social networks; nodes in many types of networks have properties that may be similar among neighbors. For example, nodes in any

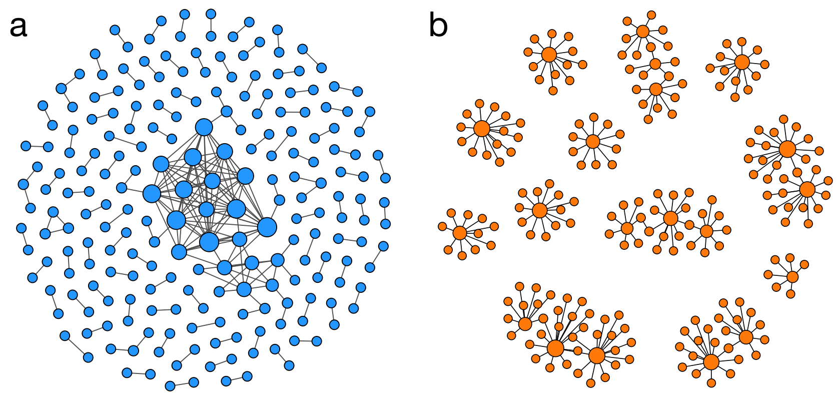


Fig. 2.2 Network degree assortativity illustrated by (a) an assortative network and (b) a disassortative network.

network have the fundamental property of degree. Assortativity based on degree is called *degree assortativity* or *degree correlation*: this occurs when high-degree nodes tend to be connected to other high-degree nodes, while low-degree nodes tend to have other low-degree nodes as neighbors. Networks with this property are called *assortative*.

An example of assortative network is shown in Figure 2.2(a): the hubs form a densely connected *core* while low-degree nodes are loosely attached to each other and/or to core nodes. Therefore we say that assortative networks have a *core-periphery structure* (discussed in greater detail in Chapter 3). Social networks are often assortative. Networks where high-degree nodes tend to be connected to low-degree nodes and vice-versa are called *disassortative*. An example is shown in Figure 2.2(b): hubs are situated at the center of star-like components. The Web, the Internet, food webs, and other biological networks tend to be disassortative.

There are two ways to measure the degree assortativity of a network, both based on measuring *correlation* between degrees of neighbor nodes. We say that two variables are *positively (negatively) correlated* if larger values of one variable tend to correspond to larger (smaller) values of the other. Pearson's correlation coefficient is a common way to measure correlation; it takes values in $[-1, +1]$ with zero meaning no correlation and ± 1 meaning perfect positive/negative correlation. One measure of network assortativity is the *assortativity coefficient*, defined as the Pearson correlation between the degrees of pairs of linked nodes. Using NetworkX:

```
r = nx.degree_assortativity_coefficient(G)
```

When the assortativity coefficient is positive, the network is assortative, and when it is negative, the network is disassortative.

The second method is based on measuring the average degree of the neighbors of a node i:

$$k_{nn}(i) = \frac{1}{k_i} \sum_j a_{ij} k_j, \tag{2.1}$$

where $a_{ij} = 1$ if i and j are neighbors, and zero otherwise. We then define the *k-nearest-neighbors* function $\langle k_{nn}(k) \rangle$ for nodes of a given degree k as the average of $k_{nn}(i)$ across all nodes with degree k. If $\langle k_{nn}(k) \rangle$ is an increasing function of k, then high-degree nodes tend to be connected to high-degree nodes, therefore the network is assortative; if $\langle k_{nn}(k) \rangle$ decreases with k, the network is disassortative. Using NetworkX we can calculate the correlation between the degree and its associated neighbor connectivity:

```python
import scipy.stats
knn_dict = nx.k_nearest_neighbors(G)
k, knn = list(knn_dict.keys()), list(knn_dict.values())
r, p_value = scipy.stats.pearsonr(k, knn)
```

Note that we need the `scipy` package for the Pearson correlation.

In Chapter 4 we will learn about a type of homophily based on information content that is very important for the Web.

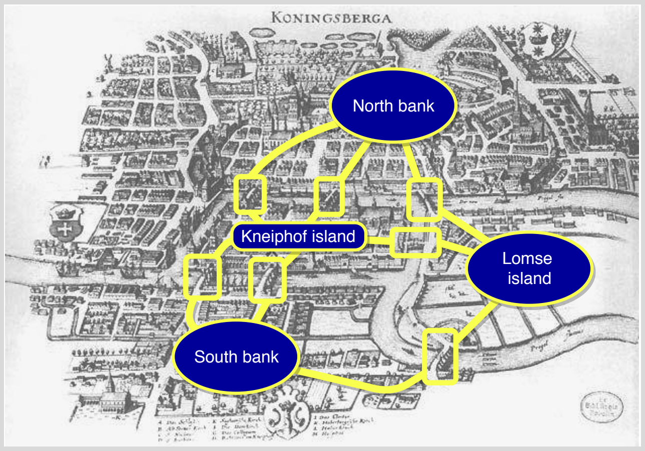
2.2 Paths and Distances

If it is possible to go from a *source* node to a *target* node by traversing links in the network, we say that there is a *path* between the two nodes. The path is the sequence of links traversed. The number of links in a path is called the *path length*. There may be multiple paths between the same two nodes. These paths may have different lengths, and may or may not share some common links. In directed paths, we must comply with link directions. A *cycle* is a special path that can be traversed to go from one node back to itself. A *simple path* never goes through the same link more than once; in this book we only focus on simple paths. Finding paths was the earliest problem studied in network science (Box 2.1).

The concept of path is at the basis of the definition of *distance* among nodes in a network. The natural distance measure between two nodes is defined as the minimum number of links that must be traversed in a path connecting the two nodes. Such a path is called the *shortest path*, and its length is called *shortest path length*. There may be multiple shortest paths between two nodes; obviously they all must have the same length. In Section 2.5 we will see how to find the shortest path between two nodes. In some cases, such as transportation networks, one can imagine that a link is associated with a geographic distance between the adjacent nodes. In such cases we can redefine the path length as the *sum of the distances*

Box 2.1 The seven bridges of Königsberg

In 1736, Leonhard Euler used graph theory to solve a mathematical problem for the first time. The Prussian city of Königsberg was divided by the Pregel river into four land masses (North and South banks, Kneiphof and Lomse islands) connected by seven bridges. The problem was to devise a walk through the city that would cross each bridge once and only once. Euler formulated a generalized version of this problem as finding a path through a network where nodes and links represent the land masses and bridges, respectively, and each link is to be traversed exactly once.



Euler proved that such a path (now called *Eulerian path* in his honor) exists only if all nodes have even degree, except the source and the target. Nodes must have even degree because for each incoming link to arrive at a node, there must be an outgoing link to depart from the node. The source and the target, if distinct, must have odd degree because when the path starts (ends) it does not "cross" the node. If they coincide (*Eulerian cycle*), there cannot be nodes with odd degree. Since all four nodes in the Königsberg network have odd degree, there was no Eulerian path in this case.

associated with the links along the path; the length of a path from Berlin to Rome by way of Paris is the sum of the the distances from Berlin to Paris and from Paris to Rome. You can think of an unweighted network as a special case in which all links have distance one.

Figure 2.3 illustrates the shortest paths between two nodes in different types of network, which depend on whether the network is directed and/or weighted. In

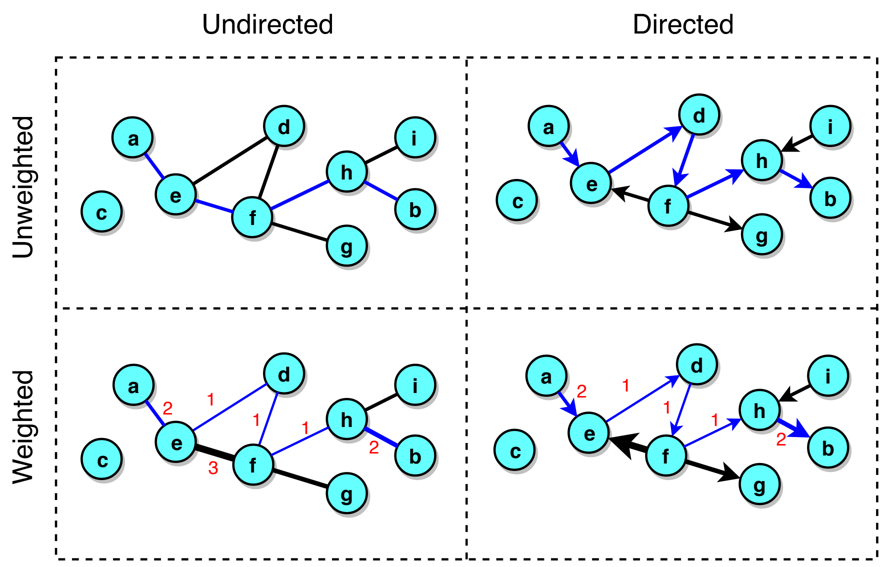


Fig. 2.3 Shortest paths in undirected, directed, unweighted, and weighted networks. Link weights represent distances and are shown in red. In each case the shortest path between nodes **a** and **b**, or from **a** to **b** in directed cases, is highlighted in blue. There is no path path between node **c** and any other node. In directed networks, the shortest path must be consistent with the direction of the links along the path; there is no directed path from **b** to **a**.

the case of an undirected, unweighted network, the shortest path is just the one that minimizes the number of links traversed, and it is the same irrespective of the direction that we move between the nodes. Note that there are two paths between nodes **a** and **b**, but the one that passes through node **d** is longer by one link; the shortest path bypasses node **d** by following the link between **e** and **f**. The shortest path length is $\ell_{ab} = 4$ links. The directed, unweighted network case is different because directed paths must be consistent with the direction of the links along the path. Therefore there is only one path from source **a** to target **b**, and it goes through **d**. The shortest directed path length is $\ell_{ab} = 5$ links. Beware that in a directed network there may be no paths between some pairs of nodes. For instance, if a node has only incoming links there are no paths having that node as the source. In the example of Figure 2.3, there are no paths from **g** to any other node. Similarly, there are no paths to nodes that have only outgoing links, such as **a**.

The undirected, weighted network in Figure 2.3 shows what happens when we use link distances. In this case the shortest path between **a** and **b** goes through **d**: it has an extra link, but the sum of the distances between **e** and **f** through **d** is 1+1=2, which is less than the distance 3 associated with the link (**e**, **f**). The directed, weighted case is straightforward: the shortest path is obtained by minimizing the sum of the distances along the path, while respecting the directions of the links. In both weighted network examples, the shortest path length is $\ell_{ab} = 7$.

In many networks, link weights express a measure of similarity or intensity of interaction between two connected nodes. We may then be interested in finding paths with large weights. A common approach is to transform the weights into distances by taking the inverse (one divided by the weight), so that a large weight corresponds to a short distance. Then the problem becomes equivalent to finding short-distance paths.

By using the shortest path length as a measure of distance among nodes, it is possible to define aggregate distance measures for an entire network: the *average shortest path length* (or simply *average path length*) is obtained by averaging the shortest path lengths across all pairs of nodes. The *diameter* of the network is instead the maximum shortest path length across all pairs of nodes, *i.e.*, the length of the longest shortest path in the network. The name is inspired by geometry, where the diameter is the longest distance between any two points on a circle.

Formally we define the *average path length* of an undirected, unweighted network as

$$\langle \ell \rangle = \frac{\sum_{i,j} \ell_{ij}}{\binom{N}{2}} = \frac{2\sum_{i,j} \ell_{ij}}{N(N-1)} \tag{2.2}$$

where ℓ_{ij} is the shortest path length between nodes i and j, and N is the number of nodes. The sum is over all pairs of nodes, and we divide by the number of pairs to compute the average. In the directed network case the definition is analogous, but the distance ℓ_{ij} is based on the shortest directed path between i and j, and each pair of nodes is considered twice for paths in both directions:

$$\langle \ell \rangle = \frac{\sum_{i,j} \ell_{ij}}{N(N-1)}. \tag{2.3}$$

The weighted cases are similar, with ℓ_{ij} defined based on link distances. The *diameter* of a network is

$$\ell_{max} = \max_{i,j} \ell_{ij}. \tag{2.4}$$

The definitions of average path length and diameter assume that the shortest path length is defined for each pair of nodes. If there are any pairs without a path, then average path length and diameter are not defined. For example, the networks in Figure 2.3 have no path between the singleton node **c** and any other node. We can think of such missing paths as paths with infinite distance. There are a few ways to deal with these cases.

If one wishes to define the average path length in a network where some of the paths do not exist, the following formula can be used for undirected networks:

$$\langle \ell \rangle = \left(\frac{\sum_{i,j} \frac{1}{\ell_{ij}}}{\binom{N}{2}} \right)^{-1} . \tag{2.5}$$

Note that if there is no path between i and j, $\ell_{ij} = \infty$ and therefore $1/\ell_{ij} = 0$ is defined. The same trick can be used for directed networks.

In Section 2.3 we show a couple of different ways to calculate the network distance and diameter when some paths are missing.

Both average path length and diameter can be used to describe the typical distance of a network. In this book we use the former. Although by definition the average cannot exceed the maximum, the two terms are sometimes used interchangeably because the two quantities behave similarly as the network size grows.

NetworkX has functions to determine the existence of paths, find shortest paths, and measure the length of a path or the average path length of a network. In the case of the undirected, unweighted network in Figure 2.3:

```
nx.has_path(G, 'a', 'c')              # False
nx.has_path(G, 'a', 'b')              # True
nx.shortest_path(G, 'a', 'b')         # ['a','e','f',h','b']
nx.shortest_path_length(G,'a','b')    # 4
nx.shortest_path(G, 'a')              # dictionary
nx.shortest_path_length(G, 'a')       # dictionary
nx.shortest_path(G)                   # all pairs
nx.shortest_path_length(G)            # all pairs
nx.average_shortest_path_length(G)    # error
G.remove_node('c')                    # make G connected
nx.average_shortest_path_length(G)    # now okay
```

When only the source node is specified, we obtain a dictionary with all shortest paths, or all shortest path lengths, from the source. When neither source nor target are given, we get an object with the shortest paths for all pairs of nodes.

For directed networks, the functions are the same but they properly account for link directions. In the case of the directed, unweighted network in Figure 2.3:

```
nx.has_path(D, 'b', 'a')        # False
nx.has_path(D, 'a', 'b')        # True
nx.shortest_path(D, 'a', 'b')   # ['a','e','d','f',h','b']
```

For weighted networks, we can store distances associated with links as weight attributes. We can then tell NetworkX to interpret the weights as distances when computing path lengths. In the case of the weighted, undirected network in Figure 2.3:

```
nx.shortest_path_length(W, 'a', 'b')             # 4
nx.shortest_path_length(W, 'a', 'b', 'weight')   # 7
```

2.3 Connectedness and Components

To relate network structure and function, it is useful to consider the *connectedness* of a network. The connectedness defines many properties of a network's physical structure. For example, in Chapter 3 it will allow us to study the robustness of a network.

Recall from Chapter 1 that the number of links in a network is bound by the number of nodes. This is an upper bound; there is no lower bound, as a network might have no links at all, uninteresting as that may be. As we shall see in Chapter 5, the higher the density, the greater the chances that the network is *connected*, *i.e.*, that you can reach any node from any other node by following a path along links and intermediate nodes. The fewer the links and lower the density, the higher the chances that the network is disconnected, so that there are multiple nodes or groups of nodes that are not reachable from each other.

NetworkX has algorithms to determine whether a network is connected. For example, the networks in Figure 1.2 are all connected:

```
K4 = nx.complete_graph(4)
nx.is_connected(K4)         # True
C = nx.cycle_graph(4)
nx.is_connected(C)          # True
P = nx.path_graph(5)
nx.is_connected(P)          # True
S = nx.star_graph(6)
nx.is_connected(S)          # True
```

If a network is not connected, we say that it is *disconnected*; it is composed of more than one *connected components* or simply *components*. A component is a subnetwork containing one or more nodes, such that there is a path connecting any pair of these nodes, but there is no path connecting them to other components. The largest connected component in many real networks includes a substantial portion of the network and is called the *giant component*. In a connected network, the giant component coincides with the entire graph.

Figure 2.4 illustrates the components of a network, that are differently defined based on undirected and directed paths in undirected and directed networks, respectively. In the undirected case, the example in the figure contains three components. Note that by definition, a singleton belongs to its own component because it is not connected to any other nodes. In the directed case, things are a bit more complicated because we have to pay attention to link directions when determining whether a node can be reached from another. We can of course ignore the link directions and treat the links as if they were undirected. In this case we refer to the components as *weakly connected*. The directed network in Figure 2.4 has three weakly connected components. However, not all nodes in a weakly connected component may be reachable from each other following *directed* paths.

In a *strongly connected* component, there is at least one directed path between

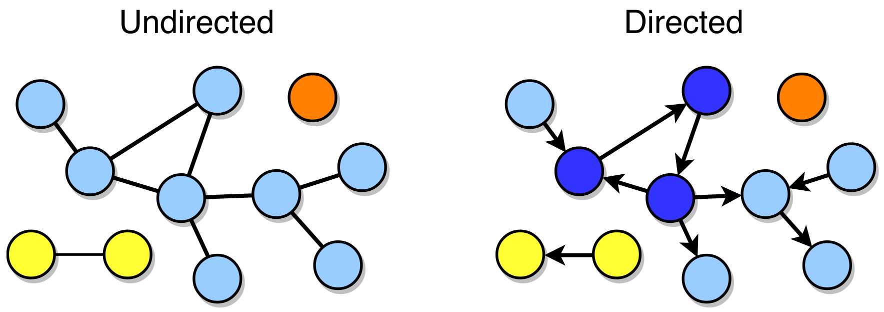


Fig. 2.4 Connected components. Colors indicate different components. In the undirected network example we observe three components, one of which is a singleton. The light blue nodes make up the giant component. In the directed example, we observe three *weakly* connected components. The largest weakly connected component contains nodes of different shades of blue; the dark blue nodes make up the largest *strongly* connected component.

every pair of nodes, in both directions. In Figure 2.4, the largest strongly connected component contains three nodes; every other node belongs to its own strongly connected component. Note that in a strongly connected network or component, there is at least one directed cycle from each node. To see why, consider any two nodes **a** and **b** in a strongly connected network. Since there must be a directed path from **a** to **b** and one from **b** to **a**, then a cycle can be constructed by combining the two.

Having defined connected components, let us return to the issue of measuring network distance when a network is disconnected. One way is to consider only the nodes in the giant component. Another approach is to average the distance only across pairs of nodes in the same component, but considering all components. To calculate the diameter of a disconnected network, one can calculate the diameter of each component and then take the maximum.

We can identify the set of nodes from which one can reach a strongly connected component S, but that cannot be reached from S — if they were, they would be part of S. This set is called the *in-component* of S. Similarly, we define the *out-component* of S as the set of nodes that can be reached from S but from which one cannot reach S.

We say that a directed network is *strongly connected* if it is a single strongly connected component. A directed network is *weakly connected* if it is a single weakly connected component.

NetworkX provides functions to identify the connected components of a network. Assume `G` and `D` are the undirected and directed networks in Figure 2.4, respectively:

```
nx.is_connected(G)                          # False
comps = sorted(nx.connected_components(G),
               key=len, reverse=True)
```

```
nodes_in_giant_comp = comps[0]
GC = nx.subgraph(G, nodes_in_giant_comp)
nx.is_connected(GC)                          # True
nx.is_strongly_connected(D)                  # False
nx.is_weakly_connected(D)                    # False
list(nx.weakly_connected_components(D))
list(nx.strongly_connected_components(D))    # lots of
                                             # singletons
```

In this example, we make use of the builtin function `sorted()` to list and sort the output of the `connected_components()` function. We specify `key=len` in order to sort by the component sizes, and `reverse=True` to output in descending order. Then the first element is the giant component.

2.4 Trees

Let us introduce a special class of undirected, connected networks such that the deletion of any one link will disconnect the network into two components. Such graphs are called *trees*.

The number of links in a tree is $L = N - 1$. To convince yourself that this is the case, start with a network with $N = 2$ nodes, which needs $L = 1$ link to be connected. Then as we add one node at a time, we must add a link to connect the new node to some existing node. So the number of links is always equal to the number of nodes minus one. Removing any link will disconnect at least one node.

Trees have other interesting properties. They have no cycles. We can prove that trees cannot have cycles by contradiction: if a tree had a cycle, we could remove at least one link of the cycle without disconnecting it. Therefore the network would not be a tree — a contradiction. Because there are no cycles, given any pair of nodes, there is only a single path connecting them.

Trees are *hierarchical*. You can pick any node in a tree and call it a *root*. Each node in a tree is connected to a parent node (toward the root) and to one or more children nodes (away from the root). The exceptions are the root, which has no parent, and the so-called *leaves* of the tree, which have no children. The hierarchical structure of trees is illustrated in Figure 2.5.

NetworkX has algorithms to determine whether a network is a tree. For example, a complete network with more than two nodes has cycles and therefore is not a tree. The star and path networks in Figure 1.2 are examples of trees:

```
K4 = nx.complete_graph(4)
nx.is_tree(K4)              # False
C = nx.cycle_graph(4)
nx.is_tree(C)               # False
```

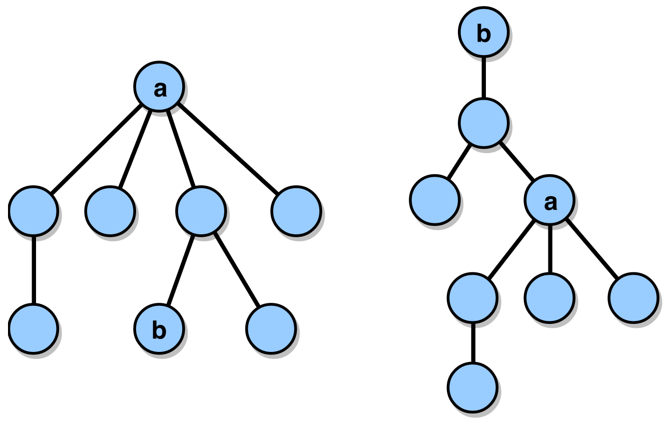


Fig. 2.5 Hierarchical structure of trees. The same tree is depicted with two different layouts, respectively with nodes **a** and **b** taken as roots and positioned at the top. Each node has its parent above (the root has no parent) and its children below (the leaves are at the bottom and have no children).

```
P = nx.path_graph(5)
nx.is_tree(P)              # True
S = nx.star_graph(6)
nx.is_tree(S)              # True
```

2.5 Finding Shortest Paths

In Section 2.2 we discussed shortest paths. But how to find the shortest path between two nodes in practice? To do this, it is necessary to map and navigate through the whole network. This is done by NetworkX and other network analysis tools. As we will see in Chapter 4, it is also done by search engines through *Web crawlers*, computer programs that automatically surf the Web finding and storing new pages.

The algorithm, or procedure for navigating through a network starting from a *source* node and finding the shortest path between the source and every other node in the network is called *breadth-first search*. The idea is that we visit the entire "breadth" of the network, within some distance from the source, before we move to a greater "depth," farther away from the source. This process is illustrated in Figure 2.6 for a simple undirected network: starting from some source node, we visit its neighbors (layer 1) and set the distance of those vertices from the source to one. We then visit the neighbors of the nodes in layer 1, except the nodes already explored (layer 2), and set their distance to two. Then we visit the neighbors of the layer 2 nodes if not previously visited (layer 3), setting their distance to three; and so on. Figure 2.6 shows that each layer contains all nodes having equal distance

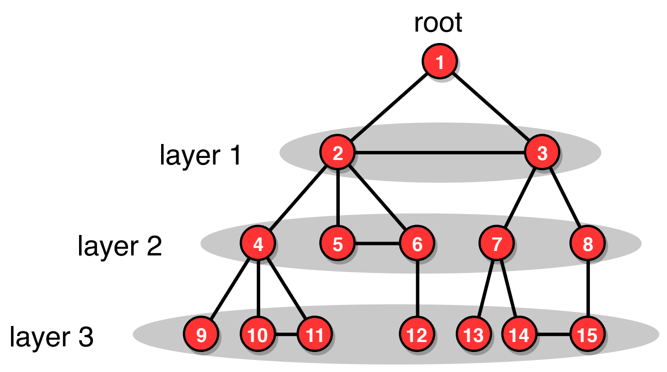


Fig. 2.6 Breadth-first search. In this instance, node **1** is selected as the source. First, we visit the neighbors of **1**, which are **2** and **3**. This is layer 1, including all nodes one step away from the source. Then we move to their neighbors **4**, **5**, **6**, **7**, **8**, which are two steps away from the source (layer 2). Finally, we reach nodes **9**, **10**, **11**, **12**, **13**, **14**, **15**, at distance three from the source (layer 3).

from the source. If the network is connected, all nodes are reached and assigned a distance from the source. The procedure is analogous for directed networks such as the Web, except that we only reach nodes through directed paths from the source.

To find the shortest paths from the source to the other nodes, the breadth-first search algorithm builds a directed *shortest-path tree*, containing the same nodes as the original network but only a subset of the links. The tree maps the shortest paths between its root (the source node) and all other nodes. The algorithm is illustrated in Figure 2.7 for a directed network; for implementation details see Box 2.2.

After the breadth-first search algorithm has executed, all nodes in the same connected component as the source node have been assigned a distance from the source. To find the shortest path from the source to any target node, we have to follow the links in the shortest-path tree backward from the target node through predecessors in the upper layers, until we arrive to the source. Recall that in a tree there is a single path to the root; each node has a single predecessor. Then we have to reverse the path to obtain the shortest path from the source to the target. In an undirected network this is the same as the path from the target to the source, but in a directed network they may be different.

In the example of Figure 2.7, we can see the shortest-path tree as its links are drawn with solid lines. For example, say we are interested in the shortest path from node **1** to node **7**. Breadth-first search has set the length of this path to $\ell_{1,7} = 2$. To find the path, we go from **7** to its predecessor in the shortest-path tree, which is node **2**, and then to its predecessor, which is the root node **1**. Reversing this path we obtain the shortest path $\mathbf{1}\to\mathbf{2}\to\mathbf{7}$. Note that this is not the only short path — the path $\mathbf{1}\to\mathbf{3}\to\mathbf{7}$ has the same length, but the algorithm only identifies *one*

Box 2.2 Breadth-First Search

Breadth-first search takes a source node as input. To implement the algorithm, each node must have an attribute used to store its distance from the source. In addition, we must keep a queue of nodes that we call *frontier*. A queue is a First-In-First-Out data structure: nodes are extracted (dequeued) in the order in which they are inserted (queued).
Initially, the source node s is queued into the frontier. Its distance is set to $\ell(s, s) = 0$, and for all other nodes the distance is set to a common unrealistic value, say -1. The network that will eventually become the shortest-path tree is initialized with no links.
In each iteration, we visit the next node i in the frontier. The node is dequeued. Then, for each successor j of i (or each neighbor if the network is undirected), we follow three steps unless j already has its distance set:

1 Queue j into the frontier.
2 Set the distances of j from the source to $\ell_{s,j} = \ell_{s,i} + 1$.
3 Add a directed link $(i \rightarrow j)$ to the shortest-path tree.

The procedure ends when the frontier is empty. If any nodes remain with unknown distance, they are not reachable from the source; they must be in a different connected component of the network.

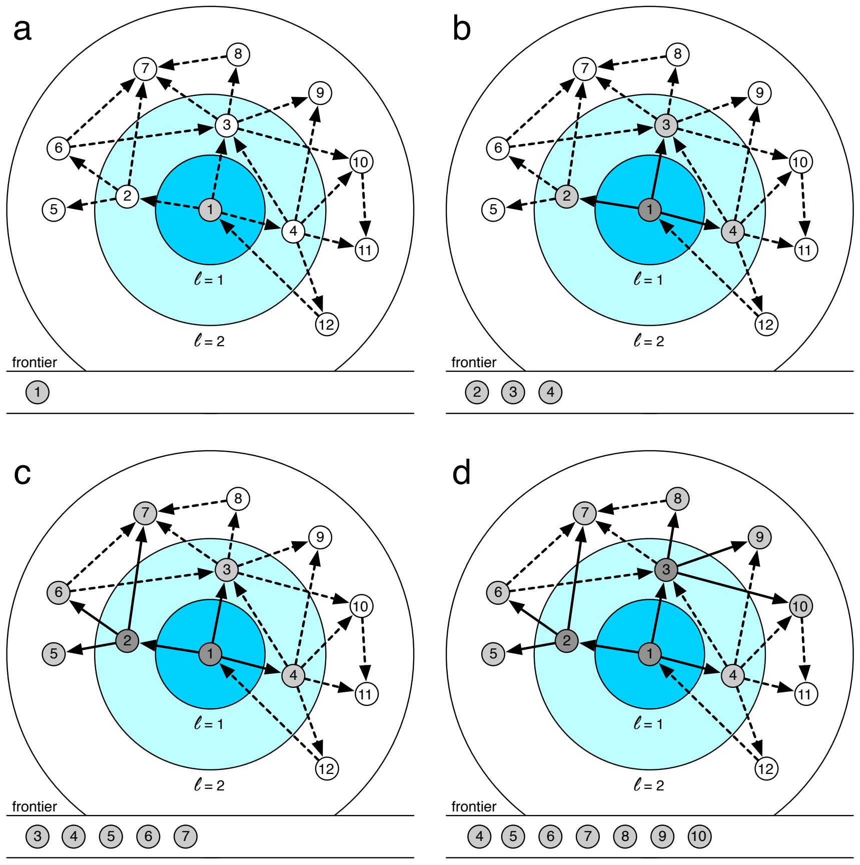


Fig. 2.7 Illustration of the breadth-first search algorithm to traverse a directed network and find short paths from a source node. Nodes are colored in light gray when they are added into the frontier queue, and dark gray when they are dequeued. Links turn from dashed to solid lines when they are added to the shortest-path tree. (a) The frontier is initialized with the source node **1**. (b) Node **1** is dequeued and its successors **2**, **3**, and **4** are queued into the frontier. (c) Node **2** is dequeued and its successors **5**, **6**, ad **7** are queued. (d) Node **3** is dequeued and its successors **8**, **9**, and **10** are queued. Node **7** is already in the frontier, therefore the link to it from node **3** is ignored. The breadth-first algorithm keeps track of visited nodes because their distances are set, so they are not queued into the frontier again. For example, when node **4** is visited, successor node **3** can be ignored because its distance from the source is already set to one. In the next step, all nodes at distance one from the source will have been visited, so that nodes at distance two can be visited.

shortest path from the source. Note also that in this directed network, the shortest path from node **7** to node **1** is not the same; in fact there is no such path.

The breadth-first search algorithm finds the shortest paths from a single source to all other nodes in an unweighted network. Slightly more complicated algorithms also exist for shortest paths in weighted networks. If we wish to find the shortest path between every pair of nodes, we need to run the algorithm N times, once from each node as a source. This is computationally expensive. In fact, as an exercise, you should try to use the NetworkX method `shortest_path(G)` (or `shortest_path_length(G)`) on some of the networks in the book's Github repository[1] (also see Table 2.1). You will notice that it takes a painfully long time for large networks; even if short paths exist, they are not necessarily easy to find. Fortunately, as we will see in Section 7.4, networks often provide us with clues so that we can search for a target node efficiently by following heuristic rules.

2.6 Social Distance

The average path length, defined in Section 2.2, characterizes how close or far we expect nodes to be in a network. Intuitively, in a grid-like network like road networks and power grids, paths can be long. Is this typical of many real-world networks? Let us start by considering a few social networks, in which this question has been explored extensively.

Coauthorship networks are a well-studied kind of social collaboration network because it is relatively easy to gather data about nodes and links. Nodes are scholars, and links can be mined from digital libraries. When we see a publication coauthored by two or more scholars, we can infer links between them in the network.

Paul Erdős was a famous mathematician who made critical contributions to network science, discussed in Chapter 5. (For more background about his life see Box 5.1.) Mathematicians are fond of studying their distance in the coauthorship network from the particular node corresponding to Erdős. They call this distance their *Erdős number* (Box 2.3). Many mathematicians have a very small Erdős number. Figure 2.8 illustrates the network of collaborations involving Erdős and his over 500 coauthors. In reality, scholars are not just close to Erdős; they are close to everyone. This is typical of collaboration networks: there are short paths among all pairs of nodes. Pick any two scholars and they will not be very far from each other.

It turns out that not only collaboration networks, but pretty much all social networks have very short paths among nodes. You are likely to know someone who knows someone who knows someone... and in a few steps you can get to anyone on the planet! For a demonstration from a more familiar domain, let us turn to the social network connecting movie stars. As we have seen in Chapter 0, nodes are actors and actresses, and two nodes are linked if they have co-starred in a movie.

[1] `github.com/CambridgeUniversityPress/FirstCourseNetworkScience`

Box 2.3 The Erdős number

Paul Erdős was one of the world's greatest mathematicians. He also stands out among scientists due to his amazing productivity and number of collaborators. Therefore Erdős plays an important role in the connectedness of the scientific collaboration network, in that one can go from many nodes of the graph to many others through him. This is so much so that a special measure has been defined in his honor: the *Erdős number*. Many scientists proudly display their Erdős number on their homepages and CVs. This number is simply defined as the length of the shortest path, in the coauthorship network, from a scholar to Paul Erdős. There is even an online tool to compute the Erdős number for mathematicians (`www.ams.org/mathscinet/collaborationDistance.html`). For example, Erdős was a collaborator of Fan Chung, who coauthored a report with Alex Vespignani, who has been a coauthor of two of the authors of this book, who therefore have an Erdős number of three. Because of the huge number of Paul Erdős' coauthors, the number of scholars with a small Erdős number is quite large.

Six degrees of Kevin Bacon is a fun game that originates from such a network. The game, illustrated in Figure 2.9, consists of finding the shortest path connecting an arbitrary actress or actor to Kevin Bacon in the co-star network. For example, a path of length $\ell = 2$ connects Marilyn Monroe to Kevin Bacon. You can play this game online at *The Oracle of Bacon* (`oracleofbacon.org`). The website pulls data to build the network from the Internet Movie Database (`IMDB.com`). While Kevin Bacon is often jokingly considered "the" hub of the star network, in reality he is not special; you can enter any pair of actors/actresses and the Oracle shows you the shortest path as a sequence of nodes (stars) and links (movies). Can you find two familiar stars separated by more that four links? Play this game and try!

The Erdős number and Oracle of Bacon demonstrate that finding long paths in real-world networks is not easy. When we think about it, the concept of short social distance — that we are all only a few steps from each other in the social network — is a familiar one. How many times have you met someone and then been surprised to discover a common friend? The low expectation to run into a friend of a friend is rooted in our intuition of how small our circle of acquaintances is, compared to an entire population. Yet this sort of thing happens often enough, prompting us to exclaim "what a small world!" A *small world* is the popular notion that social distances are short, on average. Consequently, the number of friends of friends out there is much, much larger than we think, and finding short paths in the social network is not so strange after all.

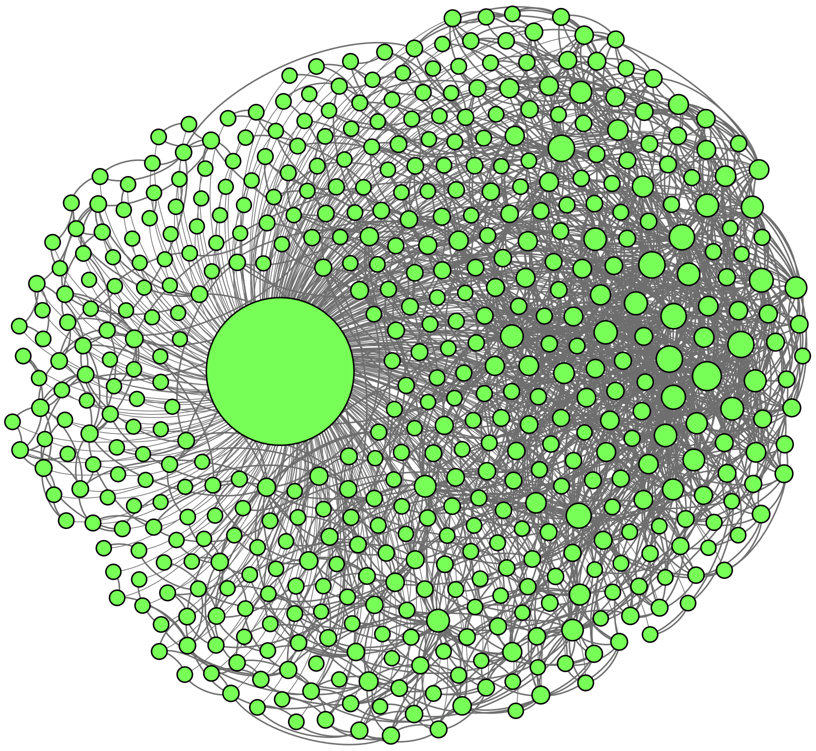


Fig. 2.8 Ego network of Paul Erdős (the large node at the center) within the the coauthorship network. Ego networks are defined in Section 1.4.

2.7 Six Degrees of Separation

The name of the game *Six degrees of Kevin Bacon* is inspired by the concept of *six degrees of separation*. The idea is the same as that of a small world: any two people in the world are connected by a short chain of acquaintances. In other words, social networks have a short diameter and an even shorter average path length. The number “six” in the expression originated from Hungarian author Frigyes Karinthy in the 1920s, and some credit goes to Italian inventor Guglielmo Marconi as well for coming up with the same idea twenty years before, in the early 1900s. However, what made the “six degrees” expression famous was an experiment conducted by

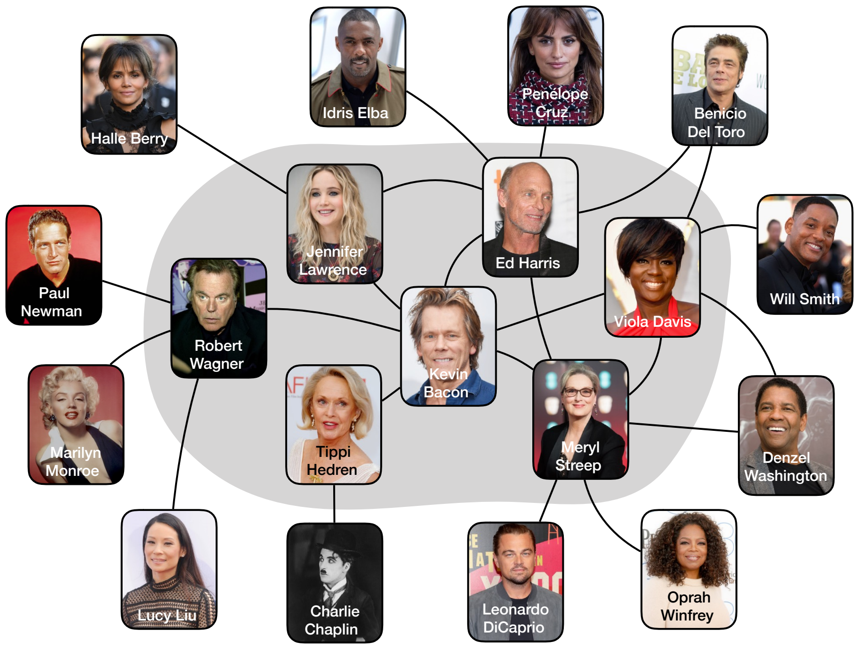


Fig. 2.9 Illustration of the *Six degrees of Kevin Bacon* game. A few of the nodes connected to Kevin Bacon in the co-star network are shown in the shaded area, along with links among them. A small sample of the nodes at distance $\ell = 2$ are also included. Photo credit: Getty Images.

psychologist Stanley Milgram[2] in the 1960s, which provided the first empirical evidence of small worlds.

Milgram wanted to measure the social distance between strangers. He therefore asked 160 subjects in Nebraska and Kansas to forward a letter to an acquaintance, with instructions that the letter should eventually reach a target person in Massachusetts. Each recipient was supposed to forward the letter to someone known, who was likely to know the target. Only 42 of the letters (26%) reached the target. In those cases, however, the path lengths were surprisingly short, ranging between 3 and 12 steps. Figure 2.10 illustrates a typical path of 4 steps. The average path length was a little more than six steps, which would eventually inspire a play titled "six degrees of separation" that ultimately popularized the small-world notion. Migram's experiment was replicated in 2003 using emails to recruit a larger number of subjects. There were 18 targets in 13 countries. Of the more than 24 thousand chains started, only 384 were completed, with an average length of four steps. When ac-

[2] Milgram is famous for another, very controversial experiment, in which subjects were instructed to inflict pain on other people. The goal was to test the degree to which a human being is capable of immoral acts as a result of pressure from authority.

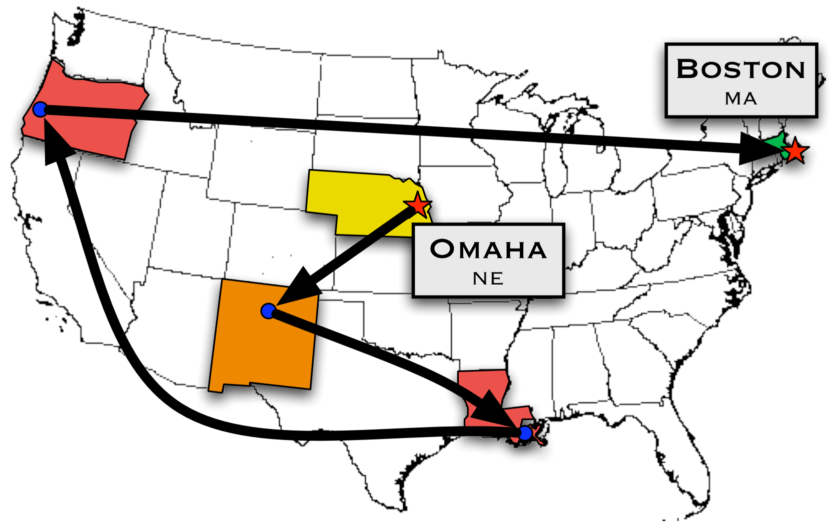


Fig. 2.10 The path followed by one of the letters in Milgram's experiment. The source subject in Omaha, Nebraska sent the letter to an acquaintance in Santa Fe, New Mexico. From there the letter was forwarded to people in New Orleans, Louisiana and Eugene, Oregon before reaching the target in Boston, Massachusetts.

counting for the many broken chains, the authors estimated a median path length of 5–7 steps, in agreement with Milgram's "six degrees." Even more recently, in 2011, researchers at Facebook and the University of Milan examined all 721 million Facebook users who were active at that time (more than 10% of the global population), with 69 billion friendships among them, and found that the average path length was 4.74 steps.

So far we have been referring to the paths we find when playing a game like *The Oracle of Bacon*, or those reported in the studies by Milgram and other researchers, as "short." But when can we call a path *short*? Compared to what? Would we call a path with six steps *short* in a network with only ten nodes? Clearly we must define what we mean by *short* paths more precisely, and the definition must be relative to the size of the network. In fact, it makes more sense to observe the relationship between the average path length $\langle \ell \rangle$ and the network size N when we consider networks (or subnetworks) of different sizes. We say that the average path length is *short* when it grows very slowly with the size of the network.

We can express slow growth mathematically by saying that the average path length scales *logarithmically* with the size of the network:

$$\langle \ell \rangle \sim \log N.$$

The logarithm of a in base b, $\log_b a$, is the exponent c such that $b^c = a$. Base $b = 10$ is commonly used; $\log_{10} 10 = 1$ because $10^1 = 10$, $\log_{10} 100 = 2$ because $10^2 = 100$, $\log_{10} 1000 = 3$, etc. The logarithm is therefore a function that grows very slowly.

What this means is that the network could have tens of millions of nodes, and yet its average path length would be in the single digits. Furthermore, the network could multiply many times in size while the average path length would only get a few steps longer.

Short paths that obey this kind of relationship are found across social networks, including academic collaborations, actor networks, networks of high school friends, and online social networks such as Facebook. Short social distances can be useful, say, when we are looking for a job. But short paths are not an exclusive feature of social networks. In fact, searching for paths is something we do routinely in all kinds of networks, for example when we book a long flight and try to minimize the number of intermediate stops. Finding network paths can be fun, too. *Wikiracing* is a hypertext search game designed to work with Wikipedia. A player must navigate from a source article to a target article, both randomly selected, solely by clicking links within each article. The goal is to reach the target in the fewest clicks, *i.e.*, to find a network path with few links. There are variations for teams and with a race against the clock. You can play several versions of this game online, such as *The Wiki Game* (`thewikigame.com`). You will be amazed at how quickly you can reach any target with a bit of practice. This tells us that the Wikipedia has short paths. The same is true for the Web, as we will see in Chapter 4.

As it turns out, short paths are a ubiquitous feature in almost all real-world networks; grid-like networks are among the few exceptions. Table 2.1 reports the average path length of various networks.[3] In all of these examples, the average path length is only a few steps. In the case of the movies and stars network the paths appear to be longer. However, keep in mind that this is a bipartite network in which a link connects a movie and an actor/actress. If we consider the co-star network, in which two stars are connected if they acted together (as in Figure 2.9), links are associated with movies; in this case the average path length is roughly cut in half. In Chapter 5 we will find short paths even in the simplest of networks, where links are assigned at random.

[3] Datasets for these networks are available in the book's Github repository: `github.com/CambridgeUniversityPress/FirstCourseNetworkScience`

Table 2.1 Average path length and clustering coefficient of various network examples. The networks are the same as in Table 1.1, their numbers of nodes and links are listed as well. Link weights are ignored. The average path length is measured only on the giant component; for directed networks we consider directed paths in the giant strongly connected component. To measure the clustering coefficient in directed networks, we ignore link directions.

Network	Nodes (N)	Links (L)	Average path length ($\langle \ell \rangle$)	Clustering coefficient (C)
Facebook Northwestern Univ.	10,567	488,337	2.7	0.24
IMDB movies and stars	563,443	921,160	12.1	0
IMDB co-stars	252,999	1,015,187	6.8	0.67
Twitter US politics	18,470	48,365	5.6	0.03
Enron Email	87,273	321,918	3.6	0.12
Wikipedia math	15,220	194,103	3.9	0.31
Internet routers	190,914	607,610	7.0	0.16
US air transportation	546	2,781	3.2	0.49
World air transportation	3,179	18,617	4.0	0.49
Yeast protein interactions	1,870	2,277	6.8	0.07
C. elegans brain	297	2,345	4.0	0.29
Everglades ecological food web	69	916	2.2	0.55

2.8 Friend of a Friend

In a social network, if Alice and Bob are both friends of Charlie's, they are also likely to be friends of each other. In other words, there is a good chance that a friend of my friend is also my friend. This translates into the presence of many *triangles* in the network. As illustrated in Figure 2.11(a), a *triangle* is a triad (set of three nodes) where each pair of nodes is connected. The connectivity among the neighbors of the nodes is an important feature of the local structure of the network because it captures how tightly knit, or *clustered*, the nodes are.

The *clustering coefficient* of a node is the *fraction of pairs of the node's neighbors that are connected to each other*. This is the same as the ratio between the number of triangles that include the node, and the maximum number of triangles in which the node *could* participate.

The *clustering coefficient* of node i is formally defined as

$$C(i) = \frac{\tau(i)}{\tau_{max}(i)} = \frac{\tau(i)}{\binom{k_i}{2}} = \frac{2\tau(i)}{k_i(k_i-1)} \qquad (2.6)$$

where $\tau(i)$ is the number of triangles involving i. The maximum possible number of triangles for i is the number of pairs formed by its k_i neighbors. Note that $C(i)$ is only defined if the degree $k_i > 1$ due to the terms k_i and $k_i - 1$ in the

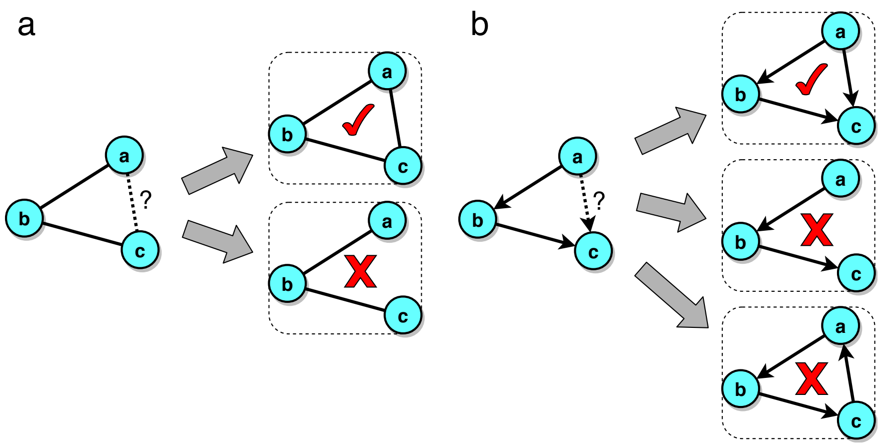


Fig. 2.11 Triads and triangles. (a) In an undirected network, node **b** has neighbors **a** and **c**. They may or may not form a triangle, depending on whether or not **a** and **c** are connected to each other. (b) In a directed network, node **a** links to **b** and **b** links to **c**. A shortcut link from **a** to **c** would form a directed triangle.

denominator: a node must have at least two neighbors for any triangle to be possible.

The clustering coefficient of the entire network is the average of the clustering coefficients of its nodes:

$$C = \frac{\sum_{i:k_i>1} C(i)}{N_{k>1}}. \tag{2.7}$$

Nodes with degree $k < 2$ are excluded when calculating the average clustering coefficient.

Figure 2.12 illustrates how to calculate the clustering coefficient for a few nodes in a network. Node **a** has two neighbors **f** and **g** that are connected to each other, forming a triangle. Therefore its clustering coefficient is $C(a) = 1/1 = 1$. Node **b** has four neighbors. Only two of the six pairs of neighbors are connected: (**e**, **c**) and (**c**, **g**). Therefore $C(b) = 2/6 = 1/3$. Node **c** has three neighbors that form two triangles via links (**e**, **b**) and (**b**, **g**). The third possible triangle is not realized because the link (**e**, **g**) is missing. Therefore $C(c) = 2/3$. Finally, node **d** has a single neighbor **e**, therefore $C(d)$ is undefined.

Our definition of clustering coefficient only applies to undirected networks, because we have only defined undirected triangles. We could extend the definition to directed networks, but it depends on the kinds of triangles that are relevant to a specific case. On Twitter, for example, we might be interested in triangles that shortcut paths along which information travels. Consider the scenario in Figure 2.11(b): if **a** follow **b** and **b** follows **c**, then **a** might be interested in following

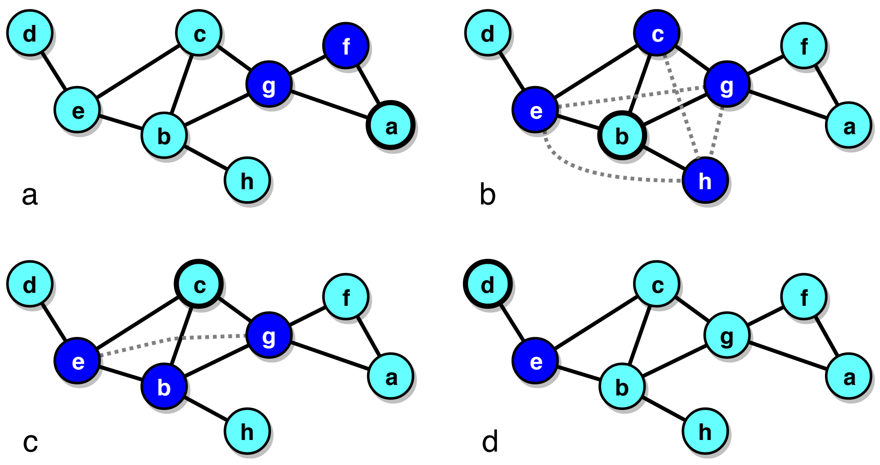


Fig. 2.12 Examples of clustering coefficient. (a) Node **a** has two neighbors **f** and **g** that are connected, forming a triangle. (b) Node **b** has four neighbors **c**, **e**, **g**, and **h**. Two of the six pairs of neighbors are connected, forming two out of six possible triangles. The missing triangle connections are shown by dotted gray lines. (c) Node **c** has three neighbors **e**, **b**, and **g** forming two out of three possible triangles. (d) Node **d** has a single neighbor **e**, therefore there are no possible triangles and the clustering coefficient is undefined.

c in order to access **c**'s posts directly rather than through **b**'s retweets. In such a scenario, we may want to only count directed triangles that encode these kinds of shortcuts. In this book we only deal with the clustering coefficient in undirected networks; for directed networks we can simply ignore the direction of the links and treat them as if they were undirected when calculating the clustering coefficient.

By averaging the clustering coefficient across the nodes, we can calculate a clustering coefficient for an entire network. A low clustering coefficient (near zero) means that the network has few triangles, while a high clustering coefficient (near one) means that the network has many triangles. Social networks have a large clustering coefficient; a significant portion of all possible triangles are present. For example, coauthorship networks tend to have clustering coefficient above 0.5. A simple mechanism explains the abundance of triangles in social networks: we meet people through shared contacts, thus closing triangles. This mechanism, called *triadic closure*, is discussed further in Chapter 5. Online social networks make suggestions based on triadic closure. For example, Facebook recommends "people you may know" based on common friends, and Twitter recommends accounts followed by your friends (accounts you follow). These recommendations result in high clustering.

Table 2.1 reports the clustering coefficient of various networks. We observe high clustering in many, but not all cases. The movies and stars network has $C = 0$; this is because the network is bipartite, therefore there can be no triangles; triangles would require links between pairs of movies or stars, which are not present in the bipartite network. If we instead examine the social network of co-stars, we find a high clustering coefficient. The Twitter retweet network also has a low $C = 0.03$. To understand why, consider that if Bob retweets Alice and Charlie retweets Bob, Twitter links both Bob and Charlie to the original author, Alice. Therefore each retweet cascade tree looks like a star. The only triangles stem from users participating in multiple stars.

NetworkX has functions to count triangles and calculate the clustering coefficient for nodes and networks. Currently NetworkX sets the clustering coefficient to zero for nodes with degree below two and includes those nodes in the average calculation.

```
nx.triangles(G)           # dict node -> no. triangles
nx.clustering(G, node)    # clustering coefficient of node
nx.clustering(G)          # dict node -> clustering coeff.
nx.average_clustering(G) # network's clustering coeff.
```

2.9 Summary

In this chapter we have learned about several features of networks: assortativity, connectedness, short paths, and clustering:

1 Assortativity is the correlation between the likelihood that two nodes are connected and their similarity. Similarity can be measured based on degree, content, location, topical interests, or any other node property. Assortativity in social networks can be due to homophily, the tendency of similar people to be connected; or to social influence, the tendency of connected people to be similar.

2 Paths are sequences of links connecting nodes in a network. The natural distance measure between two nodes is defined as the number of links traversed by the shortest connecting path. The simplest way to find a short path is the breadth-first search algorithm. The concepts of paths and distances can be extended to take into consideration link directions and weights.

3 A tree is a connected undirected network with as few links as possible. Trees have no cycles.

4 Connected components are subnetworks such that there exists a path between any two nodes in the same component, but not between two nodes in different components. In directed networks we distinguish between strongly and weakly connected components based on whether or not paths respect link directions.

5 The average path length of a network is found by averaging the shortest path lengths across all pairs of nodes in a connected network. If a network is not connected, usually only pairs of nodes in the same component are considered.

6 Most real networks have very short paths on average. This is known as the small-world property. The popular notion that social networks have six degrees of separation originated from Milgram's experiment.

7 The local clustering of a network is induced by the presence of triangles, or connected triads. For a node, the clustering coefficient measures the fraction of triangles out of the maximum possible number. For an entire network we can average the clustering coefficient across nodes. Social networks have high clustering due to friend-of-a-friend triangles.

2.10 Further Readings

The word "homophily" originates from the Greek "homós" (same) and "philia" (friendship). The concept was formulated by Lazarsfeld et al. (1954) and the presence of various forms of homophily has been observed in many studies of social networks (McPherson et al., 2001). Aiello et al. (2012) found that users with similar interests are more likely to be friends in various online social media platforms, and that similarity among users based on their profile metadata is predictive of social links. The k-nearest-neighbors connectivity and assortativity coefficient were introduced by Pastor-Satorras et al. (2001) and Newman (2002), respectively.

Researchers are increasingly studying the negative consequences of homophily. Exposure to news and information through the filter of like-minded individuals in online social network may facilitate the emergence of clustered communities in which our attention is focused toward information that we are already likely to know or agree with. These so-called "echo chambers" (Sunstein, 2001) and "filter bubbles" have been claimed to be pathological consequences of social media recommendation algorithms (Pariser, 2011) and to lead to polarization (Conover et al., 2011b) and viral misinformation (Lazer et al., 2018).

Algorithms for finding shortest paths and connected components in networks have a complicated history. The invention of breadth-first search is attributed to Zuse and Burke in a rejected 1945 Ph.D. thesis, and independently to Moore (1959). There are two famous algorithms for finding shortest paths in weighted networks: one by Dijkstra (1959); and the Bellman-Ford algorithm, published independently by Shimbel (1955), Ford Jr. (1956), Moore (1959), and Bellman (1958).

Milgram's experiment (Travers and Milgram, 1969) was repeated by Dodds et al. (2003) using emails. Backstrom et al. (2012) found that the average shortest path length on the network of Facebook friends is lower than five. Newman (2001) first studied the structure of scientific collaboration networks.

An accessible introduction to networks and their small-world and clustered structure is offered by Watts (2004). The existence of triangles in networks is also referred to as *transitivity* (Holland and Leinhardt, 1971). An early definition of network clustering coefficient was formulated by Luce and Perry (1949), while the local definition used in this book is due to Watts and Strogatz (1998).

The concept of triadic closure was introduced in a seminal paper by Granovetter (1973) and is discussed in Chapter 5. Studying data from a social media platform, Weng et al. (2013a) confirmed that triadic closure has a strong effect on link formation, but also found that shortcuts based on traffic are another key factor in explaining new links.

Exercises

2.1 Go through the Chapter 2 Tutorial on the book's Github repository.[4]

2.2 Recall that unless otherwise specified, the length of a path is the number of links contained therein. Given two nodes in an arbitrary undirected, connected graph, there must exist some shortest path between them. True or False: There may exist multiple such shortest paths.

2.3 True or False: Given any two nodes in a (undirected) tree, there exists exactly one path between those two nodes.

2.4 Consider an undirected, connected network with N nodes. What is the minimum number of links the network can have? If we do not require the network to be connected, does that minimum number of links change?

2.5 Recall that a tree of N nodes contains $N-1$ links. True or False: Any connected, undirected network of N nodes and $N-1$ links must be a tree.

2.6 True or False: Any undirected network of N nodes with at least N links must contain a cycle.

2.7 True or False: Any directed network of N nodes with at least N links must contain a cycle.

2.8 Consider the network defined by the adjacency matrix in Eq. 1.11. Are there any cycles in this network? Is it strongly connected? Weakly connected?

2.9 Consider the unweighted, undirected version of the network defined by the adjacency matrix in Eq. 1.11. Is this network a tree?

2.10 Consider the unweighted, undirected version of the network defined by the adjacency matrix in Eq. 1.11. What is this network's diameter?

2.11 If you convert a weakly-connected directed network to an undirected network, will the resulting network be connected? Explain why or why not.

2.12 Consider an arbitrary non-complete undirected network. Now add a single link. How has the number of nodes in this network's giant component changed as a result of this addition?

a. It has strictly decreased
b. It has decreased or stayed the same

[4] `github.com/CambridgeUniversityPress/FirstCourseNetworkScience`

c. It has increased or stayed the same
d. It has strictly increased

2.13 Consider the weighted directed network in Figure 2.13. Which of the following most accurately describes the connectedness of this network?

a. Strongly connected
b. Weakly connected
c. Disconnected
d. None of the above

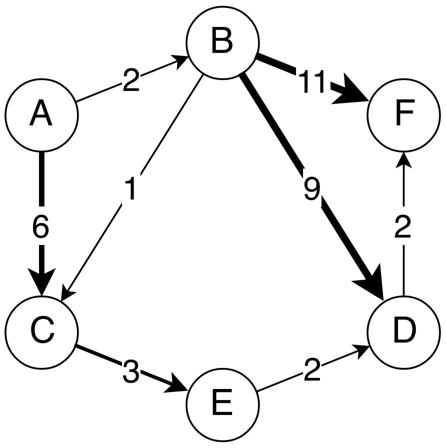


Fig. 2.13 A weighted, directed network. The numbers give the link weights.

2.14 Consider the weighted directed network in Figure 2.13. What is the in-strength of node **D**? What is the out-strength of node **C**? (Recall the definitions from Chapter 1.)

2.15 How many nodes are in the largest strongly connected component of the network in Figure 2.13?

2.16 Consider the network in Figure 2.14. Which of the following most accurately describes the connectedness of this network?

a. Strongly connected
b. Weakly connected
c. Disconnected
d. None of the above

2.17 Link weights can represent anything about the relationship between the nodes: strength of the relationship, geographic distance, voltage flowing through a link cable, etc. When discussing path lengths on a weighted graph, one must first define how the weights are related to the distances. The length of a path between two nodes is then the sum of the distances of the links in that path.

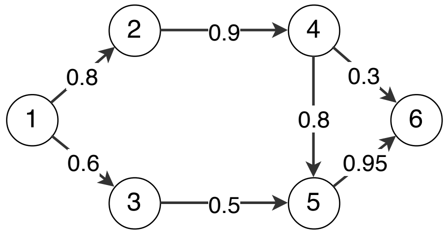


Fig. 2.14 A weighted, directed network. The numbers give the link weights.

The simplest case occurs when the link weights represent the distance. Consider the network in Figure 2.14 and assume that the link weights represent distances. Using this distance metric, what is the shortest path between nodes 1 and 6?

2.18 A common way to define the distance between two nodes is the inverse (or reciprocal) of the link weight. Consider the network in Figure 2.14, and assume that the distance between two adjacent nodes is defined as the reciprocal of the link weight. Using this distance metric, what is the shortest path between nodes 1 and 6?

2.19 Consider the network in Figure 2.15. Which of the following is the best estimate of this network's diameter?

a. 2
b. 4
c. 10
d. 20

2.20 Consider the network in Figure 2.15. Which of the following is the best estimate for the average clustering coefficient of this graph?

a. 0.05
b. 0.5
c. 0.75
d. 0.95

2.21 Would a social network be likely to have the diameter and clustering coefficient of the graph in Figure 2.15?

2.22 Consider the network in Figure 2.16. Which of the following most accurately describes the connectedness of this network?

a. Strongly connected
b. Weakly connected
c. Disconnected
d. None of the above

2.23 What is the diameter of the network in Figure 2.16?

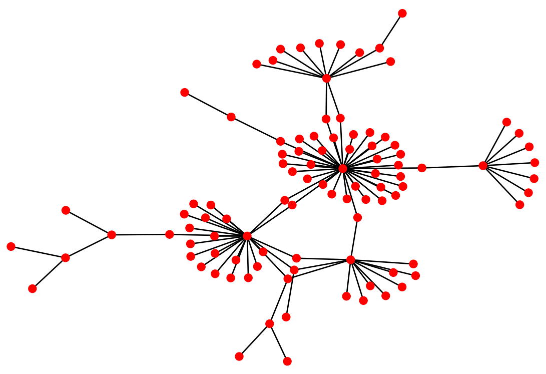


Fig. 2.15 A small sub-network of the *Drosophila melanogaster* (a.k.a. fruit fly) protein interaction network. Each node represents a protein that interacts with other proteins to perform the essential work of the cell. Experimental evidence has demonstrated that linked proteins form a molecular bond to accomplish some biological function.

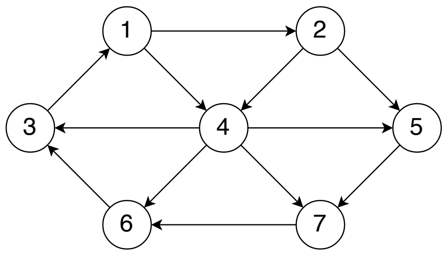


Fig. 2.16 A directed network.

2.24 Consider an undirected version of the network in Figure 2.16. What is the diameter of this network?

2.25 Consider any arbitrary directed graph D along with its undirected version G. True or False: If the average shortest path length and diameter of the directed graph exist, they can be smaller than those of the undirected version.

2.26 Imagine that you were building a competitor of NetworkX. You have already written a method `shortest_path()` to compute the shortest path between two nodes, and now you want to write a function to compute the diameter of a network. Which of the following best describes how to go about doing this?

a. First compute the shortest path lengths between each pair of nodes. The diameter is the minimum of these values.

b. First compute the shortest path lengths between each pair of nodes. The diameter is the average of these values.

c. First compute the shortest path lengths between each pair of nodes. The diameter is the maximum of these values.
d. First compute the average length of all paths between each pair of nodes. The diameter is the minimum of these values.

2.27 True or False: A network's diameter is always greater than or equal to its average path length.

2.28 What is the central idea behind the notion of "six degrees of separation"?

a. Social networks have high clustering coefficients.
b. Social networks are sparse.
c. Social networks have many high-degree nodes.
d. Social networks have small average path length.

2.29 The American Mathematical Society has a Web tool to find the *collaboration distance* between two mathematicians (see Box 2.3). Use this tool to calculate the Erdős number for a few mathematicians in your institution, or whom you know by fame.

2.30 Use *The Oracle of Bacon* (`oracleofbacon.org`) to measure the shortest-path distance in the co-star network among as many pairs of obscure actors and actresses that you can think of. Plot a histogram showing the distribution of the shortest path lengths, and also estimate the average path length based on your sample. (If you are not familiar with histograms, they are defined in the next chapter.)

2.31 Play *The Wiki Game* (`thewikigame.com`) until you are able to complete a few rounds successfully. Report the average length (numbers of clicks) of the discovered paths.

2.32 What is the maximum clustering coefficient for a node in an arbitrary undirected graph?

2.33 What is the maximum clustering coefficient for a node in a tree?

2.34 Recall the definition of ego network in Section 1.4. Consider the ego network in Figure 2.17: what is the clustering coefficient of the ego?

2.35 Consider the undirected network in Figure 2.4. Compute the shortest path length for each pair of nodes in the giant component.

2.36 Consider the undirected network in Figure 2.4. Compute the clustering coefficient for each node such that it is defined.

2.37 Consider the network example in Figure 2.12. Compute the shortest path length for each pair of nodes, and the average shortest path length for the network.

2.38 Consider the network example in Figure 2.12. Compute the clustering coefficient for each node such that it is defined, and for the network.

2.39 If you use an online social network such as Facebook or LinkedIn, measure your clustering coefficient in the network. (Hint 1: If you use a social network with directed links, such as Twitter or Instagram, you can treat the links as

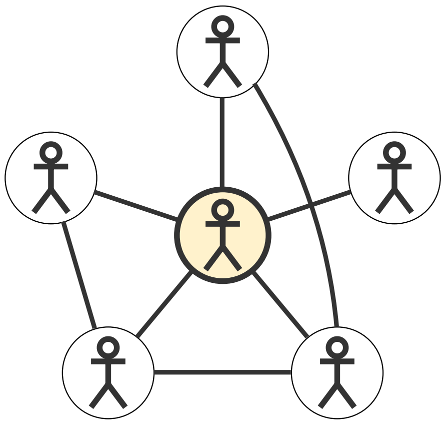


Fig. 2.17 An ego network. The ego is highlighted in yellow.

undirected.) (Hint 2: This might take a while; it's okay to make an estimate based on a small sample of your friends.)

2.40 Which of the following seemingly conflicting properties are true of social networks?

- **a.** Social networks have short paths, yet large diameter.
- **b.** Social networks have small diameter, yet large average path length.
- **c.** Social networks have many high-degree nodes, yet are disconnected.
- **d.** Social networks are highly clustered, yet are not dense.

2.41 The `socfb-Northwestern25` network in the book's Github repository is a snapshot of Northwestern University's Facebook network. The nodes are anonymous users and the links are friend relationships. Load this network into a NetworkX graph in order to answer the following questions. Be sure to use the proper graph class for an undirected, unweighted network.

1 How many nodes and links are in this network?

2 Which of the following best describes the connectedness of this network?

   - **a.** Strongly connected
   - **b.** Weakly connected
   - **c.** Connected
   - **d.** Disconnected

3 We want to obtain some idea about the average length of paths in this network, but with large networks like this it is often too computationally expensive to calculate the shortest path between every pair of nodes. If we wanted to compute the shortest path between every pair of nodes in this network, how many shortest-path calculations would be required? In other words, how many pairs of nodes are there in this network? (Hint:

Remember this network is undirected and we usually ignore self loops, especially when computing paths.)

4 To save time, let's try a sampling approach. You can obtain a random pair of nodes with

```
random.sample(G.nodes, 2)
```

Since this sampling is done without replacement, it prevents you from picking the same node twice. Do this 1000 times and for each such pair of nodes, record the length of the shortest path between them. Take the mean of this sample to obtain an estimate for the average path length in this network. Report your estimate to one decimal place.

5 Apply a slight modification to the above procedure to estimate the diameter of the network. Report the approximate diameter.

6 What is the average clustering coefficient for this network? Answer to at least two decimal places.

7 Is this network assortative or disassortative? Answer this question using the two methods shown in the text. Do the answers differ?