1 Graphs and Graph Theory

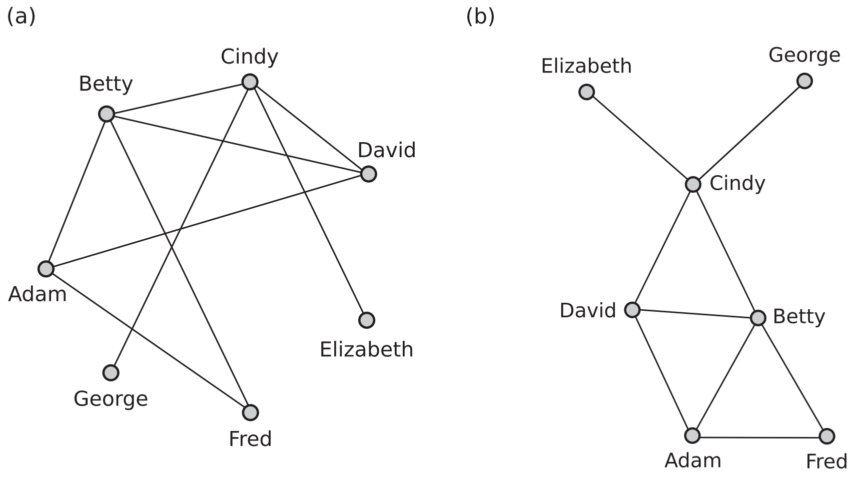
Graphs are the mathematical objects used to represent networks, and *graph theory* is the branch of mathematics that deals with the study of graphs. Graph theory has a long history. The notion of the graph was introduced for the first time in 1763 by Euler, to settle a famous unsolved problem of his time: the so-called Königsberg bridge problem. It is no coincidence that the first paper on graph theory arose from the need to solve a problem from the real world. Also subsequent work in graph theory by Kirchhoff and Cayley had its root in the physical world. For instance, Kirchhoff's investigations into electric circuits led to his development of a set of basic concepts and theorems concerning trees in graphs. Nowadays, graph theory is a well-established discipline which is commonly used in areas as diverse as computer science, sociology and biology. To give some examples, graph theory helps us to schedule airplane routing and has solved problems such as finding the maximum flow per unit time from a source to a sink in a network of pipes, or colouring the regions of a map using the minimum number of different colours so that no neighbouring regions are coloured the same way. In this chapter we introduce the basic definitions, setting up the language we will need in the rest of the book. We also present the first data set of a real network in this book, namely *Elisa's kindergarten network*. The two final sections are devoted to, respectively, the proof of the Euler theorem and the description of a graph as an array of numbers.

1.1 What Is a Graph?

The natural framework for the exact mathematical treatment of a complex network is a branch of *discrete mathematics* known as *graph theory* [48, 47, 313, 150, 272, 144]. Discrete mathematics, also called finite mathematics, is the study of mathematical structures that are fundamentally *discrete*, i.e. made up of distinct parts, not supporting or requiring the notion of continuity. Most of the objects studied in discrete mathematics are countable sets, such as integers and *finite graphs*. Discrete mathematics has become popular in recent decades because of its applications to computer science. In fact, concepts and notations from discrete mathematics are often useful to study or describe objects or problems in computer algorithms and programming languages. The concept of the graph is better introduced by the two following examples.

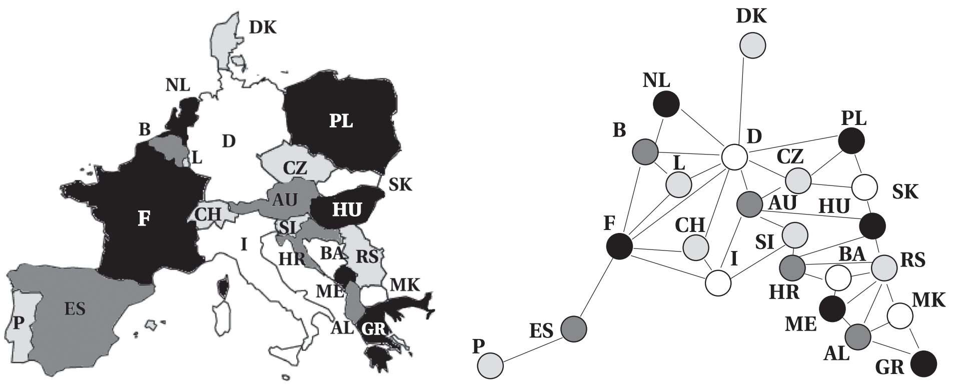
https://doi.org/10.1017/9781316216002.003 Published online by Cambridge University Press

Example 1.1 *(Friends at a party)* Seven people have been invited to a party. Their names are Adam, Betty, Cindy, David, Elizabeth, Fred and George. Before meeting at the party, Adam knew Betty, David and Fred; Cindy knew Betty, David, Elizabeth and George; David knew Betty (and, of course, Adam and Cindy); Fred knew Betty (and, of course, Adam).



The network of acquaintances can be easily represented by identifying a person by a point, and a relation as a link between two points: if two points are connected by a link, this means that they knew each other before the party. A pictorial representation of the acquaintance relationships among the seven persons is illustrated in panel (a) of the figure. Note the symmetry of the link between two persons, which reflects that if person "A" knows person "B", then person "B" knows person "A". Also note that the only thing which is relevant in the diagram is whether two persons are connected or not. The same acquaintance network can be represented, for example, as in panel (b). Note that in this representation the more "relevant" role of Betty and Cindy over, for example, George or Fred, is more immediate.

Example 1.2 *(The map of Europe)* The map in the figure shows 23 of Europe's approximately 50 countries. Each country is shown with a different shade of grey, so that from



the image we can easily distinguish the borders between any two nations. Let us suppose now that we are interested not in the precise shape and geographical position of each country, but simply in which nations have common borders. We can thus transform the map into a much simpler representation that preserves entirely that information. In order to do so we need, with a little bit of abstraction, to transform each nation into a point. We can then place the points in the plane as we want, although it can be convenient to maintain similar positions to those of the corresponding nations in the map. Finally, we connect two points with a line if there is a common boundary between the corresponding two nations. Notice that in this particular case, due to the placement of the points in the plane, it is possible to draw all the connections with no line intersections.

The mathematical entity used to represent the existence or the absence of links among various objects is called the *graph*. A graph is defined by giving a set of elements, the graph nodes, and a set of links that join some (or all) pairs of nodes. In Example 1.1 we are using a graph to represent a network of social acquaintances. The people invited at a party are the nodes of the graph, while the existence of acquaintances between two persons defines the links in the graph. In Example 1.1 the nodes of the graph are the countries of the European Union, while a link between two countries indicates that there is a common boundary between them. A *graph* is defined in mathematical terms in the following way:

Definition 1.1 (Undirected graph) *A graph, more specifically an undirected graph, $G \equiv (\mathcal{N}, \mathcal{L})$, consists of two sets, $\mathcal{N} \neq \emptyset$ and $\mathcal{L}$. The elements of $\mathcal{N} \equiv \{n_1, n_2, \ldots, n_N\}$ are distinct and are called the nodes (or vertices, or points) of the graph G. The elements of $\mathcal{L} \equiv \{l_1, l_2, \ldots, l_K\}$ are distinct unordered pairs of distinct elements of $\mathcal{N}$, and are called links (or edges, or lines).*

The number of vertices $N \equiv N[G] = |\mathcal{N}|$, where the symbol $|\cdot|$ denotes the cardinality of a set, is usually referred as the *order* of G, while the number of edges $K \equiv K[G] = |\mathcal{L}|$ is the *size* of G.[1] A node is usually referred to by a label that identifies it. The label is often an integer index from 1 to N, representing the order of the node in the set $\mathcal{N}$. We shall use this labelling throughout the book, unless otherwise stated. In an undirected graph, each of the links is defined by a pair of nodes, i and j, and is denoted as (i, j) or (j, i). In some cases we also denote the link as l_{ij} or l_{ji}. The link is said to be *incident* in nodes i and j, or to join the two nodes; the two nodes i and j are referred to as the *end-nodes* of link (i, j). Two nodes joined by a link are referred to as *adjacent* or *neighbouring*.

As shown in Example 1.1, the usual way to picture a graph is by drawing a dot or a small circle for each node, and joining two dots by a line if the two corresponding nodes are connected by an edge. How these dots and lines are drawn in the page is in principle irrelevant, as is the length of the lines. The only thing that matters in a graph is which pairs of nodes form a link and which ones do not. However, the choice of a clear drawing can be

[1] Sometimes, especially in the physical literature, the word *size* is associated with the number of nodes, rather than with the number of links. We prefer to consider K as the size of the graph. However, in many cases of interest, the number of links K is proportional to the number of nodes N, and therefore the concept of size of a graph can equally well be represented by the number of its nodes N or by the number of its edges K.

very important in making the properties of the graph easy to read. Of course, the quality and usefulness of a particular way to draw a graph depends on the type of graph and on the purpose for which the drawing is generated and, although there is no general prescription, there are various standard drawing setups and different algorithms for drawing graphs that can be used and compared. Some of them are illustrated in Box 1.1.

Figure 1.1 shows four examples of small undirected graphs. Graph G_1 is made of $N = 5$ nodes and $K = 4$ edges. Notice that any pair of nodes of this graph can be connected in only one way. As we shall see later in detail, such a graph is called a *tree*. Graphs G_2 has $N = K = 4$. By starting from one node, say node 1, one can go to all the other nodes 2, 3, 4, and back again to 1, by visiting each node and each link just once, except of course node 1, which is visited twice, being both the starting and ending node. As we shall see, we say that the graph G_2 contains a *cycle*. The same can be said about graph G_3. Graph G_3 contains an isolated node and three nodes connected by three links. We say that graphs G_1 and G_2 are connected, in the sense that any node can be reached, starting from any other node, by "walking" on the graph, while graph G_3 is not.

Notice that, in the definition of graph given above, we deliberately avoided *loops*, i.e. links from a node to itself, and *multiple edges*, i.e. pairs of nodes connected by more than one link. Graphs with either of these elements are called *multigraphs* [48, 47, 308]. An example of multigraph is G_4 in Figure 1.1. In such a multigraph, node 1 is connected to itself by a loop, and it is connected to node 3 by two links. In this book, we will deal with graphs rather than multigraphs, unless otherwise stated.

For a graph G of order N, the number of edges K is at least 0, in which case the graph is formed by N isolated nodes, and at most $N(N-1)/2$, when all the nodes are pairwise adjacent. The ratio between the actual number of edges K and its maximum possible number $N(N-1)/2$ is known as the *density* of G. A graph with N nodes and no edges has zero

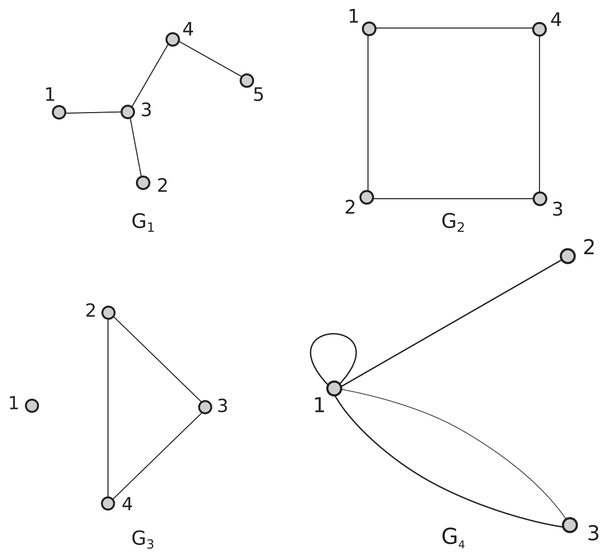
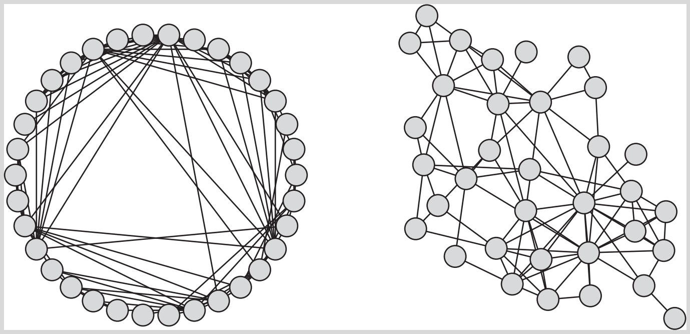


Fig. 1.1 Some examples of undirected graphs, namely a tree, G_1; two graphs containing cycles, G_2 and G_3; and an undirected multigraph, G_4.

Box 1.1 Graph Drawing

A good drawing can be very helpful to highlight the properties of a graph. In one standard setup, the so called *circular layout*, the nodes are placed on a circle and the edges are drawn across the circle. In another set-up, known as the *spring model*, the nodes and links are positioned in the plane by assuming the graph is a physical system of unit masses (the nodes) connected by springs (the links). An example is shown in the figure below, where the same graph is drawn using a circular layout (left) and a spring-based layout (right) based on the *Kamada–Kawai algorithm* [173].



By nature, springs attract their endpoints when stretched and repel their endpoints when compressed. In this way, adjacent nodes on the graph are moved closer in space and, by looking for the equilibrium conditions, we get a layout where edges are short lines, and edge crossings with other nodes and edges are minimised. There are many software packages specifically focused on graph visualisation, including *Pajek* (`http://mrvar.fdv.uni-lj.si/pajek/`), *Gephi* (`https://gephi.org/`) and *GraphViz* (`http://www.graphviz.org/`). Moreover, most of the libraries for network analysis, including *NetworkX* (`https://networkx.github.io/`), *iGraph* (`http://igraph.org/`) and *SNAP* (Stanford Network Analysis Platform, `http://snap.stanford.edu/`), support different algorithms for network visualisation.

density and is said to be *empty*, while a graph with $K = N(N-1)/2$ edges, denoted as $\mathbb{K}_N$, has density equal to 1 and is said to be *complete*. The complete graphs with $N = 3$, $N = 4$ and $N = 5$ respectively, are illustrated in Figure 1.2. In particular, $\mathbb{K}_3$ is called a *triangle*, and in the rest of this book will also be indicated by the symbol $\triangle$. As we shall see, we are often interested in the asymptotic properties of graphs when the order N becomes larger and larger. The maximum number of edges in a graph scales as N^2. If the actual number of edges in a sequence of graphs of increasing number of nodes scales as N^2, then the graphs of the sequence are called *dense*. It is often the case that the number of edges in a graph of a given sequence scales much more slowly than N^2. In this case we say that the graphs are *sparse*.

We will now focus on how to compare graphs with the same order and size. Two graphs $G_1 = (\mathcal{N}_1, \mathcal{L}_1)$ and $G_2 = (\mathcal{N}_2, \mathcal{L}_2)$ are *the same* graph if $\mathcal{N}_1 = \mathcal{N}_2$ and $\mathcal{L}_1 = \mathcal{L}_2$; that is, if both their node sets and their edge sets (i.e. the sets of unordered pairs of nodes defining $\mathcal{L}$) are the same. In this case, we write $G_1 = G_2$. For example, graphs (a) and (b) in Figure 1.3

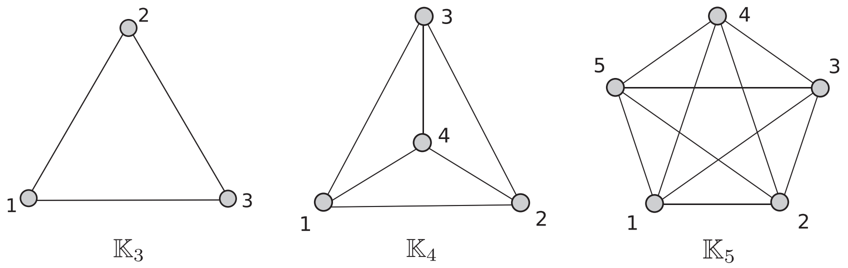


Fig. 1.2 Complete graphs respectively with three, four and five nodes.

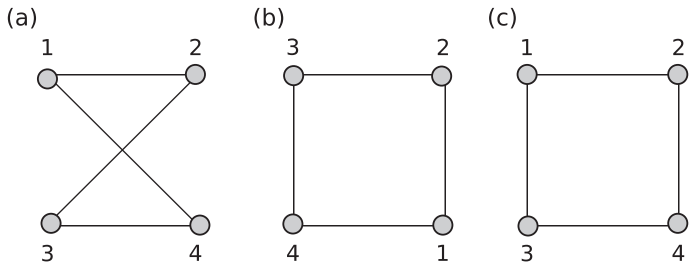


Fig. 1.3 Isomorphism of graphs. Graphs (a) and (b) are the same graph, since their edges are the same. Graphs (b) and (c) are isomorphic, since there is a bijection between the nodes that preserves the edge set.

are the same. Note that the position of the nodes in the picture has no relevance, nor does the shape or length of the edges. Two graphs that are not the same can nevertheless be *isomorphic*.

Definition 1.2 (Isomorphism) *Two graphs, $G_1 = (\mathcal{N}_1, \mathcal{L}_1)$ and $G_2 = (\mathcal{N}_2, \mathcal{L}_2)$, of the same order and size, are said to be* isomorphic *if there exists a bijection $\phi : \mathcal{N}_1 \to \mathcal{N}_2$, such that $(u, v) \in \mathcal{L}_1$ iff $(\phi(u), \phi(v)) \in \mathcal{L}_2$. The bijection ϕ is called an* isomorphism.

In other words, G_1 and G_2 are isomorphic if and only if a one-to-one correspondence between the two vertex sets $\mathcal{N}_1$, $\mathcal{N}_2$, which preserves adjacency, can be found. In this case we write $G_1 \simeq G_2$. Isomorphism is an equivalence relation, in the sense that it is reflexive, symmetric and transitive. This means that, given any three graphs G_1, G_2, G_3, we have $G_1 \simeq G_1$, $G_1 \simeq G_2 \Rightarrow G_2 \simeq G_2$, and finally $G_1 \simeq G_2$ and $G_2 \simeq G_3 \Rightarrow G_1 \simeq G_3$. For example, graph (c) in Figure 1.3 is not the same as graphs (a) and (b), but it is isomorphic to (a) and (b). In fact, the bijection $\phi(1) = 1$, $\phi(2) = 2$, $\phi(3) = 4$, and $\phi(4) = 3$ between the set of nodes of graph (c) and that of graph (a) satisfies the property required in Definition 1.2. It is easy to show that, once the nodes of two graphs of the same order are labelled by integers from 1 to N, a bijection $\phi : \mathcal{N}_1 \to \mathcal{N}_2$ can be always represented as a permutation of the node labels. For instance, the bijection just considered corresponds to the permutation of node 3 and node 4.

In all the graphs we have seen so far, a label is attached to each node, and identifies it. Such graphs are called *labelled graphs*. Sometimes, one is interested in the relation between nodes and their connections irrespective of the name of the nodes. In

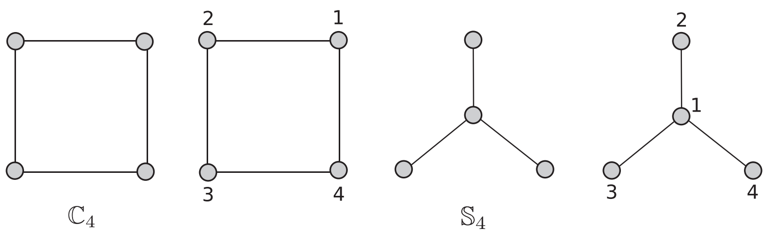


Fig. 1.4 Two unlabelled graphs, namely the cycle $\mathbb{C}_4$ and the star graph $\mathbb{S}_4$, and one possible labelling of such two graphs.

this case, no label is attached to the nodes, and the graph itself is said to be *unlabelled*. Figure 1.4 shows two examples of unlabelled graphs with $N = 4$ nodes, namely the cycle, usually indicated as $\mathbb{C}_4$, and the star graph with a central node and three links, $\mathbb{S}_4$, and two possible labellings of their nodes. Since the same unlabelled graph can be represented in several different ways, how can we state that all these representations correspond to the same graph? By definition, two unlabelled graphs are *the same* if it is possible to label them in such a way that they are the same labelled graph. In particular, if two labelled graphs are isomorphic, then the corresponding unlabelled graphs are the same.

It is easy to establish whether two labelled graphs with the same number of nodes and edges are the same, since it is sufficient to compare the ordered pairs that define their edges. However, it is difficult to check whether two unlabelled graphs are isomorphic, because there are $N!$ possible ways to label the N nodes of a graph. In graph theory this is known as the *isomorphism problem*, and to date, there are no known algorithms to check if two generic graphs are isomorphic in polynomial time.

Another definition which has to do with the permutation of the nodes of a graph, and is useful to characterise its symmetry, is that of graph *automorphism*.

Definition 1.3 (Automorphism) *Given a graph $G = (\mathcal{N}, \mathcal{L})$, an* automorphism *of G is a permutation $\phi : \mathcal{N} \to \mathcal{N}$ of the vertices of G so that if $(u, v) \in \mathcal{L}$ then $(\phi(u), \phi(v)) \in \mathcal{L}$. The number of different automorphisms of G is denoted as a_G.*

In other words, an automorphism is an isomorphism of a graph on itself. Consider the first labelled graph in Figure 1.4. The simplest automorphism is the one that keeps the node labels unchanged and produces the same labelled graph, shown as the first graph in Figure 1.5. Another example of automorphism is given by $\phi(1) = 4$, $\phi(2) = 1$, $\phi(3) = 2$, $\phi(4) = 3$. Note that this automorphism can be compactly represented by the permutation $(1, 2, 3, 4) \to (4, 1, 2, 3)$. The action of such automorphism would produce the second graph shown in Figure 1.5. There are eight distinct permutations of the labels $(1, 2, 3, 4)$ which change the first graph into an isomorphic one. The graph $\mathbb{C}_4$ has therefore $a_{\mathbb{C}_4} = 8$. The figure shows all possible automorphisms. Note that the permutation $(1, 2, 3, 4) \to (1, 3, 2, 4)$ is not an automorphism of the graph, because while $(1, 2) \in \mathcal{L}$, $(\phi(1), \phi(2)) = (1, 3) \notin \mathcal{L}$. Analogously, it is easy to prove that the number of different automorphisms of a triangle $\mathbb{C}_3 = \mathbb{K}_3$ is $a_\triangle = 6$, and more in general, for a cycle of N nodes, $\mathbb{C}_N$, we have $a_{\mathbb{C}_N} = 2N$.

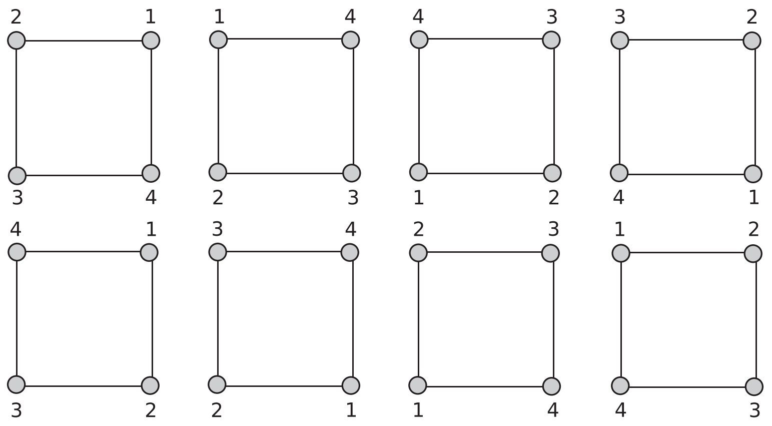


Fig. 1.5 All possible automorphisms of graph $\mathbb{C}_4$ in Figure 1.4.

Example 1.3 Consider the star graph $\mathbb{S}_4$ with a central node and three links shown in Figure 1.4. There are six automorphisms, corresponding to the following transformations: identity, rotation by 120° counterclockwise, rotation by 120° clockwise and three specular reflections, respectively around edge $(1,2)$, $(1,3)$, $(1,4)$. There are no more automorphisms, because in all permutations, node 1 has to remain fixed. Therefore, the number a_G of possible automorphisms is given by the number of permutations of the three remaining labels, that is, $3! = 6$.

Finally, we consider some basic operations to produce new graphs from old ones, for instance, by merging together two graphs or by considering only a portion of a given graph. Let us start by introducing the definition of the *union* of two graphs. Let $G_1 = (\mathcal{N}_1, \mathcal{L}_1)$ and $G_2 = (\mathcal{N}_2, \mathcal{L}_2)$ be two graphs. We define graph $G = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$ and $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$, as the *union* of G_1 and G_2, and we denote it as $G = G_1 + G_2$. A concept that will be very useful in the following is that of *subgraph* of a given graph.

Definition 1.4 (Subgraph) *A* subgraph *of $G = (\mathcal{N}, \mathcal{L})$ is a graph $G' = (\mathcal{N}', \mathcal{L}')$ such that $\mathcal{N}' \subseteq \mathcal{N}$ and $\mathcal{L}' \subseteq \mathcal{L}$. If G' contains all links of G that join two nodes in $\mathcal{N}'$, then G' is said to be the* subgraph induced *or* generated *by $\mathcal{N}'$, and is denoted as $G' = G[\mathcal{N}']$.*

Figure 1.6 shows some examples of subgraphs. A subgraph is said to be *maximal* with respect to a given property if it cannot be extended without losing that property. For example, the subgraph induced by nodes $2, 3, 4, 6$ in Figure 1.6 is the maximal complete subgraph of order four of graph G. Of particular relevance for some of the definitions given in the following is the *subgraph of the neighbours* of a given node i, denoted as G_i. G_i is defined as the subgraph induced by $\mathcal{N}_i$, the set of nodes adjacent to i, i.e. $G_i = G[\mathcal{N}_i]$. In Figure 1.6, graph (c) represents the graph G_6, induced by the neighbours of node 6.

Let $G = (\mathcal{N}, \mathcal{L})$, and let $s \in \mathcal{L}$. If we remove edge s from G we shall denote the new graph as $G' = (\mathcal{N}, \mathcal{L} - s)$, or simply $G' = G - s$. Analogously, let $\mathcal{L}' \subseteq \mathcal{L}$. We denote as

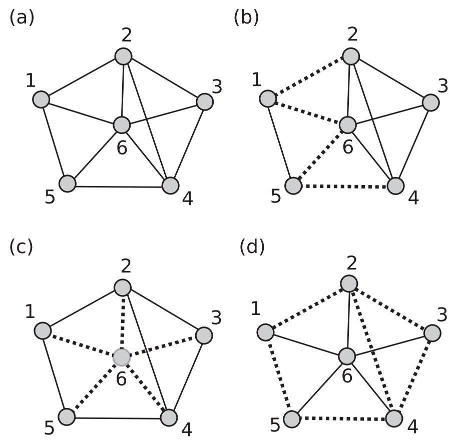


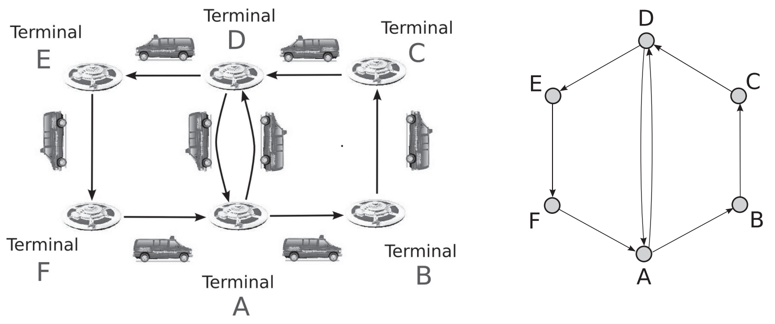
Fig. 1.6 A graph G with $N = 6$ nodes (a), and three subgraphs of G, namely an unconnected subgraph obtained by eliminating four of the edges of G (b), the subgraph generated by the set $\mathcal{N}_6 = \{1, 2, 3, 4, 5\}$ (c), and a spanning tree (d) (one of the connected subgraphs which contain all the nodes of the original graph and have the smallest number of links, i.e. $K = 5$).

$G' = (\mathcal{N}, \mathcal{L} - \mathcal{L}')$, or simply $G' = G - \mathcal{L}'$, the new graph obtained from G by removing all edges $\mathcal{L}'$.

1.2 Directed, Weighted and Bipartite Graphs

Sometimes, the precise order of the two nodes connected by a link is important, as in the case of the following example of the shuttles running between the terminals of an airport.

Example 1.4 *(Airport shuttle)* A large airport has six terminals, denoted by the letters A, B, C, D, E and F. The terminals are connected by a shuttle, which runs in a circular path, $A \to B \to C \to D \to E \to F \to A$, as shown in the figure. Since A and D are the main terminals, there are other shuttles that connect directly A with D, and vice versa. The network of connections among airport terminals can be properly described by a graph



where the $N = 6$ nodes represent the terminals, while the links indicate the presence of a shuttle connecting one terminal to another. Notice, however, that in this case it is necessary to associate a direction with each link. A directed link is usually called an arc. The graph shown in the right-hand side of the figure has indeed $K = 8$ arcs. Notice that there can be two arcs between the same pair of nodes. For instance, arc (A, D) is different from arc (D, A).

We lose important information if we represent the system in the example as a graph according to Definition 1.1. We need therefore to extend the mathematical concept of graph, to make it better suited to describe real situations. We introduce the following definition of the *directed graph*.

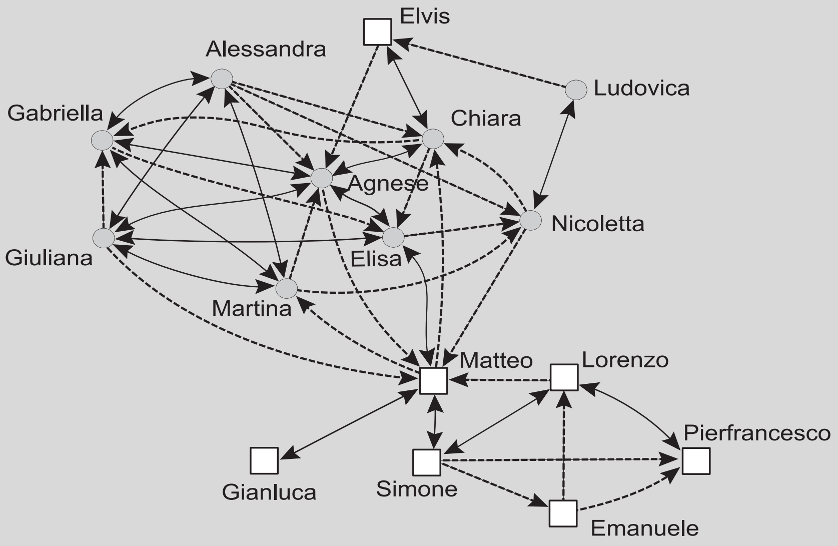
Definition 1.5 (Directed graph) *A directed graph $G \equiv (\mathcal{N}, \mathcal{L})$ consists of two sets, $\mathcal{N} \neq \emptyset$ and $\mathcal{L}$. The elements of $\mathcal{N} \equiv \{n_1, n_2, \ldots, n_N\}$ are the nodes of the graph G. The elements of $\mathcal{L} \equiv \{l_1, l_2, \ldots, l_K\}$ are distinct ordered pairs of distinct elements of $\mathcal{N}$, and are called directed links, or arcs.*

In a directed graph, an arc between node i and node j is denoted by the ordered pair (i, j), and we say that the link is *ingoing* in j and *outgoing* from i. Such an arc may still be denoted as l_{ij}. However, at variance with undirected graphs, this time the order of the two nodes is important. Namely, $l_{ij} \equiv (i, j)$ stands for an arc from i to j, and $l_{ij} \neq l_{ji}$, or in other terms the arc (i, j) is different from the arc (j, i).

As another example of a directed network we introduce here the first data set of this book, namely DATA SET 1. As with all the other data sets that will be provided and studied in this book, this refers to the network of a real system. In this case, the network describes friendships between children at the kindergarten of Elisa, the daughter of the first author of this book. The choice of this system as an example of a directed network is not accidental. Friendship networks of children are, in fact, among social systems, cases in which the directionality of a link can be extremely important. In the case under study, friendships have been determined by interviewing the children. As an outcome of the interview, friendship relations are directed, since it often happens that child A indicates B as his/her friend, without B saying that A is his/her friend. The basic properties of *Elisa's kindergarten network* are illustrated in the DATA SET Box 1.2, and the network can be downloaded from the book's webpage. Of course, one of the first things that catches our eye in the directed graph shown in Box 1.2 is that many of the relations are not reciprocated. This property can be quantified mathematically. A traditional measure of *graph reciprocity* is the ratio r between the number of arcs in the network pointing in both directions and the total number of arcs [308] (see Problem 1.2 for a mathematical expression of r, and the work by Diego Garlaschelli and Maria Loffredo for alternative measures of the reciprocity [128]). The reciprocity r takes the value $r = 0$ for a purely unidirectional graph, while $r = 1$ for a purely bidirectional one. For Elisa's kindergarten we get a value $r = 34/57 \approx 0.6$, since the number of arcs between reciprocating pairs is 34 while we have 57 arcs in total. This means that only 60 per cent of the relations are reciprocated in this network, or, more precisely, if there is an arc pointing from node i to node j, then there is a 60 per cent probability that there will also be an arc from j to i.

Box 1.2 DATA SET 1: Elisa's Kindergarten Network

Elisa's kindergarten network describes $N = 16$ children between three and five years old, and their declared friendship relations. The network given in this data set is a directed graph with $K = 57$ arcs and is shown in the figure. The nine girls are represented as circles, while the seven boys are squares. Bidirectional relations are indicated as full-line double arrows, while purely unidirectional ones as dashed-line arrows. Notice that only a certain percentage of the relations are reciprocated.



It is interesting to notice that, with the exception of Elvis, the youngest boy in the class, there is almost a split between two groups, the boys and the girls. You certainly would not observe this in a network of friendship in a high school. In the kindergarten network, Matteo is the child connecting the two communities.

Summing up, the most basic definition is that of undirected graph, which describes systems in which the links have no directionality. In the case, instead, in which the directionality of the connections is important, the directed graph definition is more appropriate. Examples of an undirected graph and of a directed graph, with $N = 7$ nodes, and $K = 8$ links and $K = 11$ arcs respectively, are shown in Figure 1.7 (a) and (b). The directed graph in panel (b) does not contain loops, nor multiple arcs, since these elements are not allowed by the standard definition of directed graph given above. Directed graphs with either of these elements are called *directed multigraphs* [48, 47, 308].

Also, we often need to deal with networks displaying a large heterogeneity in the relevance of the connections. Typical examples are social systems where it is possible to measure the strength of the interactions between individuals, or cases such as the one discussed in the following example.

Example 1.5 Suppose we have to construct a network of roads to connect N towns, so that it is possible to go from each town to any other. A natural question is: what is the

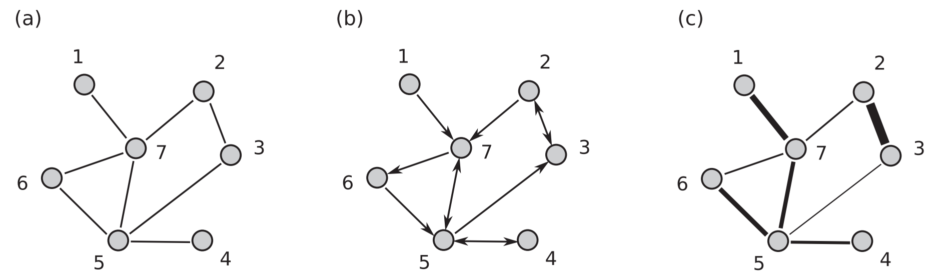


Fig. 1.7 An undirected (a), a directed (b), and a weighted undirected (c) graph with $N = 7$ nodes. In the directed graph, adjacent nodes are connected by arrows, indicating the direction of each arc. In the weighted graph, the links with different weights are represented by lines with thickness proportional to the weight.

set of connecting roads that has minimum cost? It is clear that in determining the best construction strategy one should take into account the construction cost of the hypothetical road connecting directly each pair of towns, and that the cost will be roughly proportional to the length of the road.

All such systems are better described in terms of *weighted graphs*, i.e. graphs in which a numerical value is associated with each link. The edge values might represent the strength of social connections or the cost of a link. For instance, the systems of towns and roads in Example 1.5 can be mapped into a graph whose nodes are the towns, and the edges are roads connecting them. In this particular example, the nodes are assigned a location in space and it is natural to assume that the weight of an edge is proportional to the length of the corresponding road. We will come back to similar examples when we discuss *spatial graphs* in Section 8.3. Weighted graphs are usually drawn as in Figure 1.7 (c), with the links with different weights being represented by lines with thickness proportional to the weight. We will present a detailed study of weighted graphs in Chapter 10. We only observe here that a multigraph can be represented by a weighted graph with integer weights.

Finally, a *bipartite graph* is a graph whose nodes can be divided into two disjoint sets, such that every edge connects a vertex in one set to a vertex in the other set, while there are no links connecting two nodes in the same set.

Definition 1.6 (Bipartite graph) *A bipartite graph, $G \equiv (\mathcal{N}, \mathcal{V}, \mathcal{L})$, consists of three sets, $\mathcal{N} \neq \emptyset$, $\mathcal{V} \neq \emptyset$ and $\mathcal{L}$. The elements of $\mathcal{N} \equiv \{n_1, n_2, \ldots, n_N\}$ and $\mathcal{V} \equiv \{v_1, v_2, \ldots, v_V\}$ are distinct and are called the* nodes *of the bipartite graph. The elements of $\mathcal{L} \equiv \{l_1, l_2, \ldots, l_K\}$ are distinct unordered pairs of elements, one from $\mathcal{N}$ and one from $\mathcal{V}$, and are called* links *or* edges.

Many real systems are naturally bipartite. For instance, typical bipartite networks are systems of users purchasing items such as books, or watching movies. An example is shown in Figure 1.8, where we have denoted the user-set as $U = \{u_1, u_2, \cdots, u_N\}$ and the object-set as $O = \{o_1, o_2, \cdots, o_V\}$. In such a case we have indeed only links between users and items, where a link indicates that the user has chosen that item. Notice that,

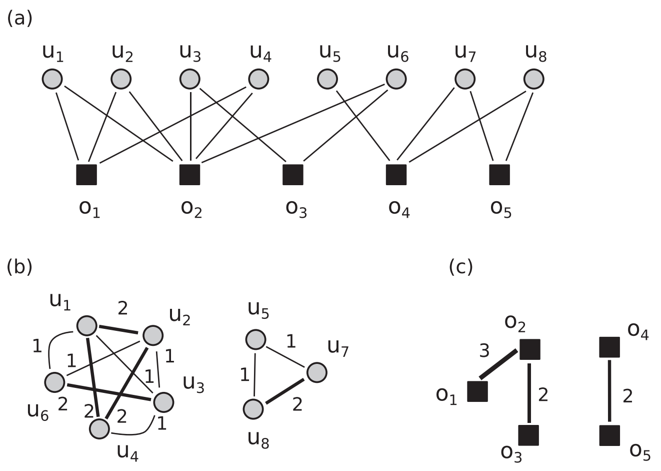


Fig. 1.8 Illustration of a bipartite network of $N = 8$ users and $V = 5$ objects (a), as well as its user-projection (b) and object-projection (c). The link weights in (b) and (c) are set as the numbers of common objects and users, respectively.

Box 1.3 Recommendation Systems

Consider a system of users buying books or selecting other items, similar to the one shown Figure 1.8. A reasonable assumption is that the users buy or select objects they like. Based on this, it is possible to construct *recommendation systems*, i.e. to predict the user's opinion on those objects not yet collected, and eventually to recommend some of them. The simplest recommendation system, known as *global ranking method* (GRM), sorts all the objects in descending order of degree and recommends those with the highest degrees. Such a recommendation is based on the assumption that the most-selected items are the most interesting for the average user. Despite the lack of personalisation, the GRM is widely used since it is simple to evaluate even for large networks. For example, the well-known *Amazon List of Top Sellers* and *Yahoo Top 100 MTVs*, as well as the list of most downloaded articles in many scientific journals, can all be considered as results of GRM. A more refined recommendation algorithm, known as *collaborative filtering* (CF), is based on similarities between users and is discussed in Example 1.13 in Section 1.6, and in Problem 1.6(c).

starting from a bipartite network, we can derive at least two other graphs. The first graph is a projection of the bipartite graph on the first set of nodes: the nodes are the users and two users are linked if they have at least one object in common. We can also assign a weight to the link equal to the number of objects in common; see panel (b) in the figure. In such a way, the weight can be interpreted as a similarity between the two users. Analogously, we can construct a graph of similarities between different objects by projecting the bipartite graph on the set of objects; see panel (c) in the figure.

1.3 Basic Definitions

The simplest way to characterise and eventually distinguishing the nodes of a graph is to count the number of their links, i.e. to evaluate their so-called *node degree*.

Definition 1.7 (Node degree) *The* degree k_i *of a node* i *is the number of edges incident in the node. If the graph is directed, the degree of the node has two components: the number of outgoing links* k_i^{out}*, referred to as the* out-degree *of the node, and the number of ingoing links* k_i^{in}*, referred to as the* in-degree *of node* i*. The total degree of the node is then defined as* $k_i = k_i^{out} + k_i^{in}$*.*

In an undirected graph the list of the node degrees $\{k_1, k_2, \ldots, k_N\}$ is called the *degree sequence*. The average degree $\langle k \rangle$ of a graph is defined as $\langle k \rangle = N^{-1} \sum_{i=1}^{N} k_i$, and is equal to $\langle k \rangle = 2K/N$. If the graph is directed, the degree of the node has two components: the average in- and out-degrees are respectively defined as $\langle k^{\text{out}} \rangle = N^{-1} \sum_{i=1}^{N} k_i^{\text{out}}$ and $\langle k^{\text{in}} \rangle = N^{-1} \sum_{i=1}^{N} k_i^{\text{in}}$, and are equal.

Example 1.6 *(Node degrees in Elisa's kindergarten)* Matteo and Agnese are the two nodes with the largest in-degree ($k^{\text{in}} = 7$) in the kindergarten friendship network introduced in Box 1.2. They both have out-degrees $k^{\text{out}} = 5$. Gianluca has the smallest in and out degree, $k^{\text{out}} = k^{\text{in}} = 1$. The graph average degree is $\langle k^{\text{out}} \rangle = \langle k^{\text{in}} \rangle = 3.6$

Another central concept in graph theory is that of the reachability of two different nodes of a graph. In fact, two nodes that are not adjacent may nevertheless be reachable from one to the other. Following is a list of the different ways we can explore a graph to visit its nodes and links.

Definition 1.8 (Walks, trails, paths and geodesics) *A* walk $W(x, y)$ *from node* x *to node* y *is an alternating sequence of nodes and edges (or arcs)* $W = (x \equiv n_0, e_1, n_1, e_2, \ldots, e_l, n_l \equiv y)$ *that begins with* x *and ends with* y*, such that* $e_i = (n_{i-1}, n_i)$ *for* $i = 1, 2, \ldots, l$*. Usually a walk is indicated by giving only the sequence of traversed nodes:* $W = (x \equiv n_0, n_1, .., n_l \equiv y)$*. The length of the walk,* $l = \ell(W)$*, is defined as the number of edges (arcs) in the sequence. A* trail *is a walk in which no edge (arc) is repeated. A* path *is a walk in which no node is visited more than once. A* shortest path *(or geodesic) from node* x *to node* y *is a walk of minimal length from* x *to* y*, and in the following will be denoted as* $\mathbb{P}(x, y)$*.*

Basically, the definitions given above are valid both for undirected and for directed graphs, with the only difference that, in an undirected graph, if a sequence of nodes is a walk, a trail or a path, then also the inverse sequence of nodes is respectively a walk, a trail or a path, since the links have no direction. Conversely, in a directed graph there might be a directed path from x to y, but no directed path from y to x.

Based on the above definitions of shortest paths, we can introduce the concept of *distance* in a graph.

Definition 1.9 (Graph distances) *In an undirected graph the* distance *between two nodes* x *and* y *is equal to the length of a shortest path* $\mathbb{P}(x, y)$ *connecting* x *and* y*. In a directed graph the distance from* x *to* y *is equal to the length of a shortest path* $\mathbb{P}(x, y)$ *from* x *to* y*.*

Notice that the definition of shortest paths is of crucial importance. In fact, the very same concept of distance between two nodes in a graph is based on the length of the shortest paths between the two nodes.

Example 1.7 Let us consider the graph shown in Figure 1.6(a). The sequence of nodes $(5, 6, 4, 2, 4, 5)$ is a walk of length 5 from node 5 back to node 5. This sequence is a walk, but not a trail, since the edge $(2, 4)$ is traversed twice. An example of a trail on the same graph is instead $(5, 6, 4, 5, 1, 2, 4)$. This is not a path, though, since node 5 is repeated. The sequence $(5, 4, 3, 2)$ is a path of length 3 from node 5 to node 2. However, this is not a shortest path. In fact, we can go from node 5 to node 2 in two steps in three different ways: $(5, 1, 2)$, $(5, 6, 2)$, $(5, 4, 2)$. These are the three shortest paths from 5 to 2.

Definition 1.10 (Circuits and cycles) *A* circuit *is a closed trail, i.e. a trail whose end vertices coincide. A* cycle *is a closed walk, of at least three edges (or arcs)* $W = (n_0, n_1, .., n_l)$, $l \geq 3$, *with* $n_0 = n_l$ *and* n_i, $0 < i < l$, *distinct from each other and from* n_0. *An undirected cycle of length* k *is usually said a* k*-cycle and is denoted as* $\mathbb{C}_k$. $\mathbb{C}_3$ *is a triangle (*$\mathbb{C}_3 = \mathbb{K}_3$*),* $\mathbb{C}_4$ *is called a quadrilater,* $\mathbb{C}_5$ *a pentagon, and so on.*

Example 1.8 An example of circuit on graph 1.6(a) is $W = (5, 4, 6, 1, 2, 6, 5)$. This example is not a path on the graph, because some intermediate vertex is repeated. An example of cycle on graph 1.6(a) is $(1, 2, 3, 4, 5, 6, 1)$. Roughly speaking a cycle is a path whose end vertices coincide.

We are now ready to introduce the concept of connectedness, first for pairs of nodes, and then for graphs. This will allow us to define what is a component of a graph, and to divide a graph into components. We need here to distinguish between undirected and directed graphs, since the directed case needs more attention than the undirected one.

Definition 1.11 (Connectedness and components in undirected graphs) *Two nodes* i *and* j *of an undirected graph* G *are said to be* connected *if there exists a path between* i *and* j. G *is said to be* connected *if all pairs of nodes are connected; otherwise it is said to be* unconnected *or* disconnected. *A* component *of* G *associated with node* i *is the maximal connected induced subgraph containing* i, *i.e. it is the subgraph which is induced by all nodes which are connected to node* i.

Of course, the first thing we will be interested in looking at, in a graph describing a real network or produced by a model, is the number of components of the graph and their sizes. In particular, when we consider in Chapter 3 families of graphs with increasing order N, a natural question to ask will be how the order of the components grows with the order of the graph. We will therefore find it useful there to introduce the definition of the *giant component*, namely a component whose number of nodes is of the same order as N.

Box 1.4 Path-Finding Behaviours in Animals

Finding the shortest route is extremely important also for animals moving regularly between different points. How can animals, with only limited local information, achieve this? Ants, for instance, find the shortest path between their nest and their food source by communicating with each other via their pheromone, a chemical substance that attracts other ants. Initially, ants explore all the possible paths to the food source. Ants taking shorter paths will take a shorter time to arrive at the food. This causes the quantity of pheromone on the shorter paths to grow faster than on the longer ones, and therefore the probability with which any single ant chooses the path to follow is quickly biased towards the shorter ones. The final result is that, due to the social cooperative behaviour of the individuals, very quickly all ants will choose the shortest path [141].

Even more striking is the fact that unicellular organisms can also exhibit similar path-finding behaviours. A well-studied case is the plasmodium of a slime mould, the *Physarum polycephalum*, a large amoeba-like cell. The body of the plasmodium contains a network of tubes, which enables nutrients and chemical signals to circulate through the organism. When food sources are presented to a starved plasmodium that has spread over the entire surface of an agar plate, parts of the organism concentrate over the food sources and are connected by only a few tubes. It has been shown in a series of experiments that the path connecting these parts of the plasmodium is the shortest possible, even in a maze [224]. Check Ref. [296] if you want to see path-finding algorithms inspired by the remarkable process of cellular computation exhibited by the *P. polycephalum*.

In a directed graph, the situation is more complex than in an undirected graph. In fact, as observed before, a directed path may exist through the network from vertex i to vertex j, but that does not guarantee that one exists from j to i. Consequently, we have various definitions of connectedness between two nodes, and we can define *weakly* and *strongly connected components* as below.

Definition 1.12 (Connectedness and components in directed graphs) *Two nodes i and j of a directed graph G are said to be* strongly connected *if there exists a path from i to j and a path from j to i. A directed graph G is said to be* strongly connected *if all pairs of nodes* (i, j) *are strongly connected. A* strongly connected component *of G associated with node i is the maximal strongly connected induced subgraph containing node i, i.e. it is the subgraph which is induced by all nodes which are strongly connected to node i.*

The undirected graph G^u *obtained by removing all directions in the arcs of G is called the* underlying undirected graph *of G. A directed graph G is said to be* weakly connected *if the underlying undirected graph* G^u *is connected. A* weakly connected component *of G is a component of its underlying undirected graph* G^u.

Example 1.9 Most graphs shown in the previous figures are connected. Examples of disconnected graphs are graph G3 in Figure 1.1 and graph (b) in Figure 1.6. Graph G3 in Figure 1.1 has two components, one given by node 1 and the other given by the subgraph induced by nodes $\{2, 3, 4\}$. Graph (b) in Figure 1.6 has also two components, one given by the subgraph generated by nodes $\{1, 5\}$ and the other generated by nodes $\{2, 3, 4, 6\}$. The directed graph in Example 1.4 is strongly connected, as it should be, since in an airport one wants to join any pair of terminals in both directions.

We will come back to component analysis and to the study of number and size of components in real-world networks in the next chapters.

1.4 Trees

Trees are a particular kind of graph that appear very commonly both in the analysis of other graphs and in various applications. Trees are important because, among all connected graphs with N nodes, are those with the smallest possible number of links. Usually a *tree* is defined as a connected graph containing no cycles. We then say that a tree is a connected *acyclic* graph. The simplest possible non-trivial tree is a graph with two nodes and two links, known as a *triad*, and usually indicated in this book with the symbol $\wedge$. Triads and triangles play an important role in complex networks, and we will come back to them in Section 4.3.

Together with the concept of the tree, we can also introduce that of the *forest*, that is, a graph whose connected components are all trees. Various examples of trees are shown in Figure 1.9. The first graph is a tree with $N = 17$ nodes. The second graph is one of the possible spanning trees of a graph with $N = 7$ nodes and $K = 12$ links. A *spanning tree* of a graph G is a tree that contains all the nodes of G, i.e. a connected subgraph which contains all the nodes of the original graph and has the smallest number of links. Finally, the third graph is a sketch of a *Cayley tree*, an infinite tree in which each node is connected to z neighbours, where z is called the coordination number. Namely, we plot the first three iterations to construct a Cayley tree with $z = 3$, starting with an origin node placed in the centre of the figure. Even if the concept of the tree graph is not familiar to you, you are bound to be familiar with many examples.

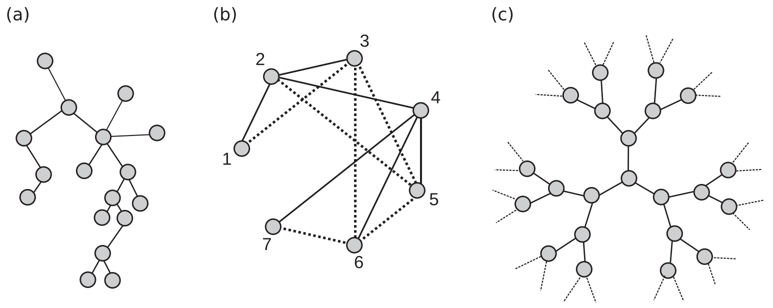


Fig. 1.9 Three examples of trees. A tree with $N = 17$ nodes (a). A spanning tree (solid lines) of a graph with $N = 7$ nodes and 12 links (solid and dashed lines) (b). Three levels of a Cayley tree with $z = 3$ (c).

Example 1.10 *(Trees in the real world)* Any printed material and textbook that is divided into sections and subsections is organised as a tree. Large companies are organised as trees, with a president at the top, a vice-president for each division, and so on. Mailing addresses, too, are trees. To send a mail we send it first to the correct country, then to the correct state, and similarly to the correct city, street and house number. There are abundant examples of trees on computers as well. File systems are the canonical example. Each drive is the root of an independent tree of files and directories (folders). File systems also provide a good example of the fact that any node in a tree can be identified by giving a path from the root. In nature, plants and rivers have a tree-like structure.

In practice, there are several equivalent ways to characterise the concept of tree, and each one can be used as a definition. Below we introduce the three definitions most commonly found in the literature.

Definition 1.13 (Trees) *A tree can be alternatively defined as:* (A) *a connected acyclic graph;* (B) *a connected graph with $K = N - 1$ links;* (C) *an acyclic graph with $K = N - 1$ links.*

The three definitions can be proven to be equivalent. Here, we shall only prove that definition (A) implies definition (B), i.e. that $(A) \Rightarrow (B)$, while we will defer the discussion of $(B) \Rightarrow (C)$ and $(C) \Rightarrow (A)$ to Problem 1.4(a). In order to show that definition (A) implies definition (B), we first need to prove the following three propositions, which also show other interesting properties of trees.

Proposition 1.1 *Let $G = (\mathcal{N}, \mathcal{L})$ be a tree, i.e. by using definition (A), a connected acyclic graph. Then, for any pair of vertices $x, y \in \mathcal{N}$ there exists one and only one walk that joins x and y.*

Proof We will provide a proof by contradiction. Let $x, y \in \mathcal{N}$. Since G is connected, there is at least one path that joins x and y. Let us assume that there exist two paths, denoted by $w_1 = (x_0 = x, x_1, x_2, \ldots, x_r = y)$ and $w_2 = (y_0 = x, y_1, y_2, \ldots, y_s = y)$. Let us denote by $u < \min(r, s)$ the largest index for which $x_i = y_i$. The two walks will reconnect for some indices j and j', i.e. it will be

$$x_j = y_{j'}, \quad x_i \neq y_{i'}, \forall i \in \{u+1, \ldots j-1\}, i' \in \{u+1, \ldots j'-1\}.$$

As shown in Figure 1.10, it follows that there exists a cycle $\mathbb{C} = (x_u, x_{u+1}, \ldots, x_j = y_{j'}, y_{j'-1}, \ldots, y_u = x_u)$, which contradicts the assumption that G is acyclic. □

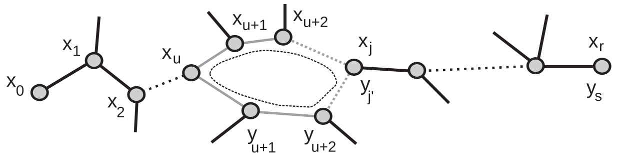


Fig. 1.10 In a tree there cannot exist two different paths that join two nodes, otherwise a cycle would form.

Proposition 1.2 *Let $G = (\mathcal{N}, \mathcal{L})$ be a graph. Suppose that for each pair of distinct nodes of the graph there exists one and only one path joining the nodes. Then G is connected and if we remove any edge $\ell \in \mathcal{L}$, the resulting graph $G - \ell$ will not be connected.*

Proof That G is connected follows immediately from the assumptions. Furthermore, if ℓ is an edge that joins x and y, since there is only one path joining two edges (from proposition 1.1), if we remove it there will be no path joining x and y, and therefore the resulting graph will be disconnected. □

Proposition 1.3 *Let G be a connected graph, such that if $\ell \in \mathcal{L} \Rightarrow G - \ell$ is disconnected. Then G is connected and has $K = N - 1$ links.*

Proof We only need to prove that $K = N - 1$. We will do this by induction on N. For $N = 1$ and $N = 2$ one has respectively $K = 0$ and $K = 1$. Now let G be a graph with $N \geq 3$, and let $x, y \in \mathcal{N}$, $(x, y) \in \mathcal{L}$, $x \neq y$. By assumption, $G - (x, y)$ is not connected: it is in fact formed by two connected components, G_1 and G_2, having respectively N_1 and N_2 nodes, with $N = N_1 + N_2$. Because $N_1 < N$, $N_2 < N$, by induction one has $N_1 = K_1 + 1$ and $N_2 = K_2 + 1$. From $K = K_1 + K_2 + 1$ it follows that

$$N = N_1 + N_1 = K_1 + 1 + K_2 + 1 = K + 1.$$

□

Finally, it is clear that by the successive use of the three propositions above we have proved that definition (A) implies definition (B).

1.5 Graph Theory and the Bridges of Königsberg

As an example of the powerful methods of graph theory, in this section we discuss the theorem proposed by the Swiss mathematician Leonhard Euler in 1736 as a solution to the Königsberg bridge problem. This is an important example of how the abstraction of graph theory can prove useful for solving practical problems. It is also historically significant, since Euler's work on the Königsberg bridges is often regarded as the birth of graph theory. The problem is related to the ancient Prussian city of Königsberg (later, the city was taken over by the USSR and renamed Kaliningrad), traversed by the Pregel river. The city, with its seven bridges, as there were in Euler's time, is graphically shown in the left-hand side of Figure 1.11. The problem to solve is whether or not it is possible to find an optimum stroll that traverses each of the bridges exactly once, and eventually returns to the starting point.

A brute force approach to this problem consists in starting from a side, making an exhaustive list of possible routes, and then checking one by one all the routes. In the case that no route satisfies the requested condition, one has to start again with a different initial point and to repeat the procedure. Of course, such an approach does not provide a general solution to the problem. In fact, if we want to solve the bridges' problem for a different city, we should repeat the enumeration for the case under study. Euler came up with an elegant

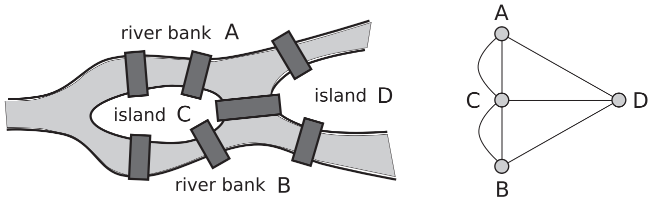


Fig. 1.11 The city of Königsberg at the time of Leonhard Euler (left). The river is coloured light grey, while the seven bridges are dark grey. The associated multigraph, in which the nodes corresponds to river banks and islands, and the links represents bridges (right).

way to answer the question for any given configuration of bridges. First, he introduced the idea of the graph. He recognised that the problem depends only on the set of connections between riverbanks and islands. If we collapse the whole river bank A to a point, and we do the same for river bank B and for the islands C and D, all the relevant information about the city map can, in fact, be encapsulated into a graph with four nodes (river banks and islands) and seven edges (the bridges) shown in right-hand side of Figure 1.11. The graph is actually a multigraph, but this will not affect our discussion. In graph terms, the original problem translates into the following request: "Is it possible to find a circuit (or trail) containing all the graph edges?" Such a circuit (or trail) is technically called an *Eulerian circuit* (*Eulerian trail*).

Definition 1.14 (Eulerian circuits and trails) *A trail in a graph G containing all the edges is said to be an Eulerian trail in G. Similarly, a circuit in G containing all the edges is said to be an Eulerian circuit in G. A graph is said to be Eulerian if it contains at least one Eulerian circuit, or semi-Eulerian if it contains at least one Eulerian trail.*

Example 1.11 *(Difficulty of an exhaustive search)* A way to perform an exhaustive search for an Eulerian trail in a graph with N nodes and K edges is to check among the walks of length $l = K$ whether there is one containing all the edges of the graph. If there is such a walk, then it is necessarily a trail, and therefore it is an Eulerian trail. The number of walks of length $l = K$ is thus a measure of the difficulty of an exhaustive search. This number can be calculated exactly in the case of the multigraph in Figure 1.11, although here we will only give an approximate estimate. Let us consider first the simpler case of a complete graph with N nodes, $\mathbb{K}_N$. The total number of walks of length l for such a graph is $N(N-1)^l$. In fact, we can choose the initial node in N different ways and, at each node, we can choose any of its $N-1$ edges. In conclusion, to look for Eulerian trails in $\mathbb{K}_N$ we have to check $N(N-1)^{N(N-1)/2}$ walks. This number is equal to 2916 in a complete graph with $N = 4$ nodes. The same argument applies to a regular graph $\mathbb{R}_{N,k}$, i.e. a graph with N nodes and k links for each node. In such a case we can choose the initial node in N different ways and, at each node, we can choose k edges. Finally, the number of walks of length l is equal to Nk^l, so that if we set $l = K$ we obtain Nk^K walks of length K. By using such a formula with k replaced by $\langle k \rangle = 2K/N$, we can get an estimate for the number of walks of length K in a generic graph with N nodes and K links. This gives 25736 for the graph of

Königsberg, having $\langle k \rangle = 3.5$. This number is of the same order of magnitude as the exact value (see Problem 1.5). Notice that this number grows exponentially with K, so that in a city with a larger number of bridges it can become impossible to explore all the different trips. For instance, in the case of the historical part of Venice, with its 428 bridges, even by assuming a small value $\langle k \rangle = 2$, we get a number of $N \cdot 2^{428}$ walks to check. Thus, an exhaustive search for an Eulerian path over the graph represented by the islands of Venice and its bridges will be far beyond the computational capabilities of modern computers. In fact, even assuming that a computer can check 10^{15} walks per second, it would be able to check about 10^{32} walks in a timespan equal to the age of the universe; this number is much smaller than $N \cdot 2^{428} \approx 7N \cdot 10^{128}$.

After having shown that the problem can be rephrased in terms of a graph, Euler gave a general theorem on the conditions for a graph to be Eulerian.

Theorem 1.1 (Euler theorem) *A connected graph is Eulerian iff each vertex has even degree. It has a Eulerian trail from vertex i to vertex j, $i \neq j$, iff i and j are the only vertices of odd degree.*

To be more precise, Euler himself actually proved only a necessary condition for the existence of an Eulerian circuit, i.e. he proved that if some nodes have an odd degree, then an Eulerian trail cannot exist. The proof given by Euler can be summarised as follows.

Proof Suppose that there exists an Euler circuit. This means that each node i is a crossing point, therefore if we denote by p_i the number of times the node i is traversed by the circuit, its degree has to be $k_i = 2p_i$, and therefore it has to be even. If we only assume the existence of an Eulerian trail, then there is no guarantee that the starting point coincides with the ending point, and therefore the degree of such two points may be odd. □

Euler believed that the converse was also true, i.e. that if all nodes have an even degree then there exists an Eulerian circuit, and he gave some argument about this, but he never rigorously proved the sufficient condition [105]. The proof that the condition that all nodes have even degree is sufficient for the existence of an Eulerian trail appeared more than a century later, and was due to the German mathematician Carl Hierholzer, who published the first characterisation of Eulerian graphs in 1873 [152]. The early history of graph theory, including the work of Euler and Hierholzer, is illustrated in [245].

Here we shall give a complete proof of the Euler theorem based on the concept of partition of a graph into cycles. Consider the set of edges $\mathcal{L}$ of a graph G. We say that a subset $\mathcal{L}_1 \subseteq \mathcal{L}$ is a cycle if there exists a cycle, Z_1, that contains all and only the edges $\mathcal{L}_1$. We say that the set $\mathcal{L}$ is *partitioned* if there exists a certain number s of subsets of $\mathcal{L}$, $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_s$, such that:

$$\mathcal{L}_i \cap \mathcal{L}_j = \emptyset,\ \forall i,j \in [1,\ldots,s], \quad \cup_{i=1}^{s} \mathcal{L}_i = \mathcal{L}$$

Now we can state the characterisation of Eulerian graphs in the form of equivalence of the following three statements [150]:

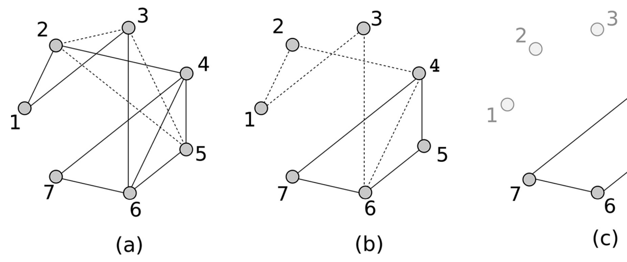


Fig. 1.12 The reduction process to prove that a graph with all nodes with even degree can be partitioned into cycles.

(1) G is an Eulerian graph (i.e. it contains at least one Eulerian circuit)
(2) $\forall i \in \mathcal{N}$, k_i is even
(3) there exists a partition of $\mathcal{L}$ into cycles.

Proof Proof (1) $\Longrightarrow$ (2). It is clear that the existence of an Eulerian circuit implies that every node i is a crossing point, i.e. every node can be considered both a "starting point" and an "ending point", therefore its degree k_i has to be even. □

Proof Proof (2) $\Longrightarrow$ (3). From property (2) it follows that in G there exists at least one cycle, Z_1, otherwise G would be a tree, and it would therefore have vertices of degree one (see Section 1.4). If $G \simeq Z_1$,[2] then (3) is proved. If $G \not\simeq Z_1$, let $G_2 = G_1 - \mathcal{L}_1$, i.e. G_2 is the graph obtained from G after removing all edges of Z_1. It is clear that all vertices of G_2 have even degree, because the degree of each node belonging to Z_1 has been decreased by 2. Let G_2' be the graph obtained from G_2 by eliminating all isolated vertices of G_2. Since all vertices of G_2' have even degree, this means that G_2 contains at least a cycle, Z_2, and the argument repeats. Proceeding with the reduction, one will at last reach a graph $G_\ell' \simeq Z_\ell$, therefore the set of links $\mathcal{L}$ is partitioned in cycles $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_\ell$. The procedure is illustrated in Figure 1.12. □

Proof Proof of (3) $\Longrightarrow$ (1). We now assume that the set of edges $\mathcal{L}$ can be partitioned into a certain number s of cycles, $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_s$. Let us denote by $Z_1, Z_2, \dots, Z_s$ the corresponding graphs. If $Z_1 \simeq G$ then (1) is proved. Otherwise, let Z_2 be a cycle with a vertex i in common with Z_1. The circuit that starts in i and passes through all edges of Z_1 and Z_2 contains all edges of Z_1 and Z_2 exactly once. Hence, it is an Eulerian circuit for $Z_1 \cup Z_2$. If $G \simeq Z_1 \cup Z_2$ the assert is proved, otherwise let Z_3 be another cycle with a vertex in common with $Z_1 \cup Z_2$, and so on. By iterating the procedure, one can construct in G an Eulerian circuit. □

The Euler theorem provides a general solution to the bridge problem: the request to pass over every bridge exactly once can be satisfied if and only if the vertices with odd degree are zero (starting and ending point coincide) or two (starting and ending point do not coincide). Now, if we go back to the graph of Königsberg we see that the conditions of the theorem are not verified. Actually, all the four vertices in the graph in Figure 1.11 have

[2] The symbol $\simeq$ indicates that the two graphs are isomorphic. See Section 1.1

an odd degree. Therefore Eulerian circuits and trails are not possible. In the same way, by a simple and fast inspection, we can answer the same question for the city of Venice or for any other city in the world having any number of islands and bridges.

1.6 How to Represent a Graph

Drawing a graph is a certainly a good way to represent it. However, when the number of nodes and links in the graph is large, the picture we get may be useless because the graph can look like an intricate ball of wool. An alternative representation of a graph, which can also prove useful when we need to input a graph into a computer program, can be obtained by using a matrix. Matrices are tables of numbers on which we can perform certain operations. The space of matrices is a vector space, in which, in addition to the usual operations on vector spaces, one defines a matrix product. Here and in Appendices A.4 and A.5 we will recall the basic definitions and operations that we will need in the book. More information can be found in any textbook on linear algebra.

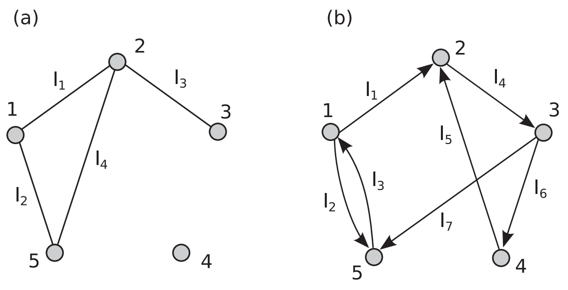
There are different ways to completely describe a graph $G = (\mathcal{N}, \mathcal{L})$ with N nodes and K links by means of a matrix. One possibility is to use the so-called *adjacency* matrix A.

Definition 1.15 (Adjacency matrix) *The* adjacency *matrix A of a graph is a $N \times N$ square matrix whose entries a_{ij} are either ones or zeros according to the following rule:*

$$a_{ij} = \begin{cases} 1 & \textit{iff } (i,j) \in \mathcal{L} \\ 0 & \textit{otherwise} \end{cases}$$

In practice, for an undirected graph, entries a_{ij} and a_{ji} are set equal to 1 if there exists the edge (i,j), while they are zero otherwise. Thus, in this case, the adjacency matrix is symmetric. If instead the graph is directed, $a_{ij} = 1$ if there exists an arc from i to j. Notice that in both cases it is common convention to set $a_{ii} = 0, \forall i = 1, \ldots, N$.

Example 1.12 Consider the two graphs in the figure below. The first graph is undirected and has $K = 4$ links, while the second graph is directed and has $K = 7$ arcs. The adjacency



matrices associated with the two graphs are respectively:

$$A_u = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}, \quad A_d = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

A_u is symmetric and contains $2K$ non-zero entries. The number of ones in row i, or equivalently in column i, is equal to the degree of vertex i. The adjacency matrix of a directed graph is in general not symmetric. This is the case of A_d. This matrix has K elements different from zero, and the number of ones in row i is equal to the number of outgoing links k_i^{out}, while the number of ones in column i is equal to the number of ingoing links k_i^{in}.

Clearly, Definition 1.15 refers only to undirected and directed graphs without loops or multiple links, which will be our main interest in this section and in the rest of the book. While weighted graphs will be treated in detail in Chapter 10, here we want to add that it is also possible to describe a bipartite graph by means of a slightly different definition of the adjacency matrix than that given above, and which in general is not a square matrix. In fact, a bipartite graph such as that shown in Figure 1.8 can be described by an $N \times V$ *adjacency matrix* A, such that entry $a_{i\alpha}$, with $i = 1, \dots, N$ and $\alpha = 1, \dots, V$, is equal to 1 if node i of the first set and node α of the second set are connected, while it is 0 otherwise. Notice that we used Roman and Greek letters to avoid confusion between the two types of nodes. Using this representation of a bipartite graph in terms of an adjacency matrix, we show in the next example how to formally describe a commonly used method to recommend a set of objects to a set of users (see also Box 1.3).

Example 1.13 *(Recommendation systems: collaborative filtering)* Consider a bipartite graph of users and objects such as that shown in Figure 1.8, in which the existence of the link between node i and node α denotes that user u_i has selected object o_α. A famous personalised recommendation system, known as *collaborative filtering* (CF), is based on the construction of a $N \times N$ *user similarity matrix* $S = \{s_{ij}\}$. The similarity between two users u_i and u_j can be expressed in terms of the adjacency matrix of the graph as:

$$s_{ij} = \frac{\sum_{\alpha=1}^{V} a_{i\alpha} a_{j\alpha}}{\mathtt{min}\{k_{u_i}, k_{u_j}\}}, \tag{1.1}$$

where $k_{u_i} = \sum_{\alpha=1}^{V} a_{i\alpha}$ is the degree of user u_i, i.e. the number of objects chosen by u_i [324]. Based on the similarity matrix S, we can then construct an $N \times V$ *recommendation matrix* $R = \{r_{i\alpha}\}$. In fact, for any user–object pair u_i, o_α, if u_i has not yet chosen o_α, i.e. if $a_{i\alpha} = 0$, we can define a recommendation score $r_{i\alpha}$ measuring to what extent u_i may like o_α, as:

$$r_{i\alpha} = \frac{\sum_{j=1, j\neq i}^{N} s_{ij} a_{j\alpha}}{\sum_{j=1, j\neq i}^{N} s_{ij}}. \tag{1.2}$$

At the numerator, we sum the similarity between user u_i and all the other users that have chosen object o_α. In practice, we count the number of users that chose object o_α, weighting each of them with the similarity with user u_i. The normalisation at the denominator guarantees that $r_{i\alpha}$ ranges in the interval $[0, 1]$. Finally, in order to recommend items to a user u_i, we need to compute the values of the recommendation score $r_{i\alpha}$ for all objects $o_\alpha, \alpha = 1, \dots, V$, such that $a_{i\alpha} = 0$. Then, all the non-zero values of $r_{i\alpha}$ are sorted in decreasing order, and the objects in the top of the list are recommended to user u_i.

Let us now come back to the main issue of this section, namely how to represent an undirected or directed graph. An alternative possibility to the adjacency matrix is a $N \times K$ matrix called the *incidence* matrix, in which the rows represent different nodes, while the columns stand for the links.

Definition 1.16 (Incidence matrix) *The incidence matrix B of an undirected graph is an $N \times K$ matrix whose entry b_{ik} is equal to* 1 *whenever the node i is incident with the link l_k, and is zero otherwise. If the graph is directed, the adopted convention is that the entry b_{ik} of B is equal to* 1 *if arc k points to node i, it is equal to* -1 *if the arc leaves node i, and is zero otherwise.*

Example 1.14 *(Incidence matrix)* The incidence matrices respectively associated with the undirected and directed graphs considered in Example 1.12 are:

$$B_u = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \quad B_d = \begin{pmatrix} -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Notice that B_u is a 5×4 matrix because the first graph has $N = 5$ nodes and $K = 4$ links, while B_d is a 5×7 matrix because the second graph has $N = 5$ nodes and $K = 7$ arcs. Also, notice that there are only two non-zero entries in each column of an incidence matrix.

Observe now that many elements of the adjacency and of the incidence matrix are zero. In particular, if a graph is sparse, then the adjacency matrix is sparse: for an undirected (directed) graph the number $2K$ (K) of non-zero elements is proportional to N, while the total number of elements of the matrix is N^2. When the adjacency matrix is sparse, it is more convenient to use a different representation, in which only the non-zero elements are stored. The most commonly used representation of sparse adjacency matrices is the so-called *ij-form*, also known as *edge list form*.

Definition 1.17 (Edge list) *The edge list of a graph, also known as ij-form of the adjacency matrix of the graph, consists of two vectors* $\mathbf{i}$ *and* $\mathbf{j}$ *of integer numbers storing the positions, i.e. respectively the row and column indices of the ones of the adjacency matrix*

A. Each of the two vectors has M components, with $M = 2K$ for undirected graphs, and $M = K$ for directed graphs.

Notice that storing the indices of the ones of the adjacency matrix is perfectly equivalent to storing the edges, i.e. the ordered pairs (i_k, j_k), $k = 1, \ldots, M$, with $M = 2K$ for undirected graphs, and $M = K$ for directed graphs. The two vectors $\mathbf{i}$ and $\mathbf{j}$ and the total number N of nodes are enough to completely represent a graph whose nodes are identified by integer numbers from 1 to N.

Example 1.15 *(ij-form of the adjacency matrix)* Below we give the *ij*-form of the adjacency matrices, A_u and A_d, of the two graphs in Example 1.12.

$$A_u = (\mathbf{i}, \mathbf{j}),\ \mathbf{i} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 2 \\ 3 \\ 5 \\ 5 \end{pmatrix},\ \mathbf{j} = \begin{pmatrix} 2 \\ 5 \\ 1 \\ 3 \\ 5 \\ 2 \\ 1 \\ 2 \end{pmatrix}, \quad A_d = (\mathbf{i}, \mathbf{j}),\ \mathbf{i} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 3 \\ 3 \\ 4 \\ 5 \end{pmatrix},\ \mathbf{j} = \begin{pmatrix} 2 \\ 5 \\ 3 \\ 4 \\ 5 \\ 2 \\ 1 \end{pmatrix}$$

For an undirected graph, such a representation is somehow redundant, because (i, j) and (j, i) represent the same edge, and it is enough to store it only once, or, in other words, in a symmetric matrix it is sufficient to store only the non-zero elements (i, j), with, say, $j \leq i$. However, we shall usually adopt the redundant representation, which allows a more uniform treatment of undirected and directed graphs. In a directed graph, the same representation of the adjacency matrix can be used. In this case the number of non-zero elements of the matrix is equal to the number of edges K. Note that the order in which the edges are stored is irrelevant: to change it is equivalent to relabelling the edges, but the graph (and the edge labels as a pair of nodes) is unchanged. We have chosen to store the pairs in such a way that the index i is increasing. This means that we first represent the edges starting from node 1, then the edges starting from node 2, and so on.

The *ij*-form of the adjacency matrix is very flexible, because the order in which the links are stored is not relevant. This form is often used during the construction of a graph which is obtained by adding a new link at a time. However, this form does not reveal straight away the neighbours of a node, so it is not the best choice in tasks such as finding the possible walks on the graph. More suitable graph representations in such cases are the *list of neighbours*, and the *compressed row storage* of the adjacency matrix. A list of neighbours stores for each node i its label, and the labels of the nodes to which i is connected. If the graph is directed we have two possibilities, namely we can list either the out-neighbours or the in-neighbours of each node. The lists of out- and in-neighbours are equivalent, although it can be convenient to choose one or the other according to the problem we face.

Example 1.16 *(List of neighbours)* The lists of neighbours corresponding to the two graphs in Example 1.12 are respectively:

$$A_u = \left(\begin{array}{c|lll} \text{node} & \text{in-neighbours} & & \\ 1 & 2 & 5 & \\ 2 & 1 & 3 & 5 \\ 3 & 2 & & \\ 4 & & & \\ 5 & 1 & 2 & \end{array} \right), \quad A_d = \left(\begin{array}{c|ll} \text{node} & \text{out-neighbours} & \\ 1 & 2 & 5 \\ 2 & 3 & \\ 3 & 4 & 5 \\ 4 & 2 & \\ 5 & 1 & \end{array} \right)$$

Notice that in the case of a directed graph, we have two different possibilities. Namely, for each node i, we can list either the nodes pointed by i, or the nodes pointing to i. In particular, in the matrix A_d given above we have adopted the first choice.

The *compressed row storage*, instead, consists of an array $\mathbf{j}$ of size $2K$ (K for directed graphs) storing the ordered sequence of the neighbours of all nodes, and a second array, $\mathbf{r}$, of size $N + 1$. The k_1 neighbours of node 1 are stored in $\mathbf{j}$ at positions $1, 2, \dots, k_1$, the k_2 neighbours of node 2 are stored in $\mathbf{j}$ at positions $k_1 + 1, k_1 + 2, \dots, k_1 + k_2$, and so on. The value of r_i, for $i = 1, 2, \dots, N$, is the index of $\mathbf{j}$ where the first neighbour of node i is stored, while r_{N+1} is equal to the size of $\mathbf{j}$ plus one ($r_{N+1} = 2K + 1$ for undirected graphs, while $r_{N+1} = K + 1$ for directed graphs). With the definitions given above, we have that $r_{i+1} - r_i$ is equal to the degree of node i if the graph is undirected, or to the out-degree of node i if the graph is directed. For a directed graph, an alternative possibility is to use the *compressed column storage*, where, for each node i, we list all the nodes pointing to i.

Example 1.17 *(Compressed storage)* The vectors $\mathbf{r}$ and $\mathbf{j}$ corresponding to matrix A_u are $\mathbf{r} = (1, 3, 6, 7, 7, 9)$ and $\mathbf{j} = (2, 5, 1, 3, 5, 2, 1, 2)$. Note that the first element of the vector $\mathbf{r}$ is always 1, while the degree k_i of node i is given by $k_i = r_{i+1} - r_i, i = 1, \dots, N$. Therefore, the $k_1 = 2$ neighbours of node 1 are stored in $\mathbf{j}$ in the positions from $r_1 = 1$ to $r_2 - 1 = 2$, the $k_2 = 3$ neighbours of node 2 are stored in $\mathbf{j}$ in the positions from $r_2 = 3$ to $r_3 - 1 = 5$, and so on. In the case of the directed graph A_d, we have two different lists of neighbours of a node, namely the out-neighbours and the in-neighbours. Consequently the compressed row storage is different from the compressed column storage. The vectors $\mathbf{r}$ and $\mathbf{j}$ corresponding to matrix A_d in the compressed row storage are $\mathbf{r} = (1, 3, 4, 6, 7, 8)$ and $\mathbf{j} = (2, 5, 3, 4, 5, 2, 1)$. Note that the out-degree of node i is $k_i^{\text{out}} = r_{i+1} - r_i$. Instead in the compressed column storage the vectors are $\mathbf{r} = (1, 2, 4, 5, 6, 8)$ and $\mathbf{j} = (5, 1, 4, 2, 3, 1, 3)$, and now $r_{i+1} - r_i$ gives the in-degree k_i^{in} of node i.

In practice, for the implementation of graph algorithms given in the Appendix, we shall make use of either the ij-form or the compressed row storage form. The details of the implementation of basic matrix operations in such forms can be found in Appendix A.4.

1.7 What We Have Learned and Further Readings

In this chapter we have introduced the basic language and the fundamental definitions we will use throughout the book. We started from scratch with the very definition of graph, providing the reader with all the different variations of the concept, namely that of *undirected*, *directed*, *weighted* and *bipartite graph*. All of these will be necessary to appropriately describe the richness and variegated nature of real-world networks. We then moved on to the most basic of node properties, the *degree*, and to some properties of pairs of nodes, such as that of *connectedness*, and we also analysed the different ways in which we can explore a graph by visiting its nodes and links. Finally, we briefly discussed how to describe a graph in terms of a matrix or a list of edges. We have also introduced the first data set of a real network of the book, that of children friendships at *Elisa's kindergarten*.

In conclusion, the main message learnt in this chapter is that *graph theory*, a well-developed branch of mathematics with a long tradition and an extended research literature, is the first pillar of complex networks. As an example of the power of graph theory we discussed how Leonhard Euler solved in 1736 the Königsberg bridge problem. This is an important historical example, since it was the first time that a practical problem was solved by transforming it into a graph problem. We will come back to a couple of other useful mathematical results from graph theory in Chapter 3, when we will be studying the properties of large random graphs. However, for a complete introduction to the field of graph theory, we suggest here the books by Douglas B. West [313], Frank Harary [150] and Béla Bollobás [48, 47]. The reader can find more on the history of graph theory in Ref. [245].

Problems

1.1 What Is a Graph?

(a) Consider the geographical map of the USA and draw the graph showing which states have a common border. In what ways do this graph and the one considered in Example 1.2 differ from the graph of social acquaintances in Example 1.1?

(b) The so-called *Four Colour Theorem* demonstrates that any map on a plane can be coloured by using only four colours, so that no two adjacent countries have the same colour [313]. Verify that this is true for the geographical map of Europe and for that of the USA.

(c) The regular solids or regular polyhedra, also known as the Platonic solids, are convex polyhedra with equivalent faces composed of congruent convex regular polygons. There are exactly five such solids, namely the tetrahedron (or triangular pyramid), the hexahedron (or cube), the octahedron, the dodecahedron and the icosahedron, as was proved by Euclid in the last proposition of the Elements.

Consider the connectivity among the vertices of each of the five regular solid polyhedra and construct the associated graphs.

1.2 Directed, Weighted and Bipartite Graphs

(a) Construct the friendship network of your class and show it as a graph, as was done for Elisa's kindergarten network in Box 1.2.

(b) Find an expression of the graph reciprocity r (introduced in Section 1.2) in terms of the adjacency matrix A of the graph.

(c) Consider the adjacency matrix:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.3}$$

Is the corresponding graph directed or undirected? Draw the graph.

(d) Consider the $N \times V$ adjacency matrix with $N = 4$ and $V = 6$:

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{1.4}$$

Draw the corresponding bipartite graph. Construct the $N \times N$ matrix $B = AA^\top$ that describes the projection on the first set of nodes. What is the meaning of the off-diagonal terms and that of the diagonal terms in such a matrix? Express, in terms of matrices A and $A^\top$, the $V \times V$ matrix C that describes the projection of the bipartite graph on the second set of nodes.

1.3 Basic Definitions

(a) Imagine you have organised a party with $N = 5$ people. Some of the people at the party know each other already, so that we focus on the party friendship network. In particular, two of them have one friend each at the party, two of them have two friends each, and one has got three friends. Is this network possible? Why? HINT: Prove that in any graph the number of nodes with odd degree must be even.

(b) Is it possible to have a party at which no two people have the same number of friends? Can you support your answer by a mathematical proof?

(c) Consider a bipartite graph with N nodes of the first type and V nodes of the second type. Prove that $\langle k \rangle_{\mathcal{N}} = \frac{V}{N} \langle k \rangle_{\mathcal{V}}$, where $\langle k \rangle_{\mathcal{N}}$ and $\langle k \rangle_{\mathcal{V}}$ are respectively the average degree of nodes of the first and second type.

(d) Prove that a graph with no odd-length cycles is bipartite.

1.4 Trees

(a) In Section 1.4 we have shown that definition (A) implies definition (B). Prove, now, by contradiction, that definition (B) implies definition (C), i.e. that $(B) \Rightarrow$

(C). Finally, prove that $(C) \Rightarrow (A)$. In this way we have shown the perfect equivalence among the three definitions of a tree (A), (B) and (C), given in Definition 1.13.

(b) Prove the following statement: If an acyclic graph G is a tree, and $(x, y) \notin \mathcal{L}$, $\Rightarrow \exists$ cycle in $G + (x, y)$. This is another possible definition of trees equivalent to definitions (A), (B) and (C) given in Section 1.4.

(c) Evaluate the number of components in a *forest* with N nodes and $K = N - c$ links.

1.5 Graph Theory and the Bridges of Königsberg

(a) The Königsberg bridge problem can be stated in terms of a graph instead of a multigraph. We can in fact construct a graph in which the nodes represent each river bank and island, but also each bridge. Show that, as expected, the obtained graph admits neither Euler circuit nor Euler trail.

(b) Show that the total number of walks of length l in a graph described by an adjacency matrix A is: $\sum_{i,j}(A^l)_{ij}$.

(c) Let us denote by A_l, B_l, C_l, D_l the number of walks of length l starting, respectively, from nodes A, B, C and D, in the multigraph of Figure 1.11. Show that the following iterative rules hold:

$$\begin{aligned} A_{l+1} &= 2C_l + D_l \\ B_{l+1} &= 2C_l + D_l \\ C_{l+1} &= 2A_l + 2B_l + D_l \\ D_{l+1} &= A_l + B_l + C_l \end{aligned}$$

with $A_0 = B_0 = C_0 = D_0 = 1$. Use a computer to show that the exact number of walks of length 7 is equal to 32554.

(d) As in Problem 1.7(c), consider the five regular polyhedra, namely the tetrahedron, the cube, the octahedron, the dodecahedron and the icosahedron. Can you trace a path along all of the edges of any of these polyhedra without going over any edge twice?

1.6 How to Represent a Graph

(a) Write the incidence matrix and the list of neighbours for the graph described by the adjacency matrix in Eq. (1.3).

(b) Write the *ij*-form of the adjacency matrix in Eq. (1.3).

(c) Consider the bipartite graph with $N = 4$ and $V = 6$ corresponding to the $N \times V$ matrix A given in Eq. (1.4), and imagine the graph describes how N users have selected V objects. Suppose, now, you want to recommend objects to users by means of the CF method. Use Eqs. (1.1) and (1.2) to compute the recommendation score for each user.