1

A First Encounter with Graphs

1.1. A few definitions

There is not much fun in listing basic definitions about graphs (this is quite a bad introduction to start with!) but if we seek a rigorous presentation of results and proofs, then we cannot avoid giving accurate definitions of the objects that we will manipulate, but hopefully nice examples will also come quickly. In this book, we assume that the reader has a basic (or, at least a naive) knowledge of sets and operations on them.

As usual in mathematics, a *pair* (u, v) made up of two elements is implicitly assumed to be ordered: it has a first component u and a second component v. It has to be compared with a *set* with two elements u and v denoted by $\{u, v\}$. A set does not carry any ordering information about its elements. In particular, if $u \neq v$, then we can build two pairs but a single set: $(u, v) \neq (v, u)$ and $\{u, v\} = \{v, u\}$. If S is a finite set, we will write $\#S$ to denote the number of elements in S, i.e. the cardinality of S. We can also find the notation $|S|$ but we will use it to denote lengths of paths.

1.1.1. *Directed graphs*

DEFINITION 1.1.– *Let V be an arbitrary set. A* directed graph*, or* ***digraph****, is a pair $G = (V, E)$ where E is a subset of the Cartesian product $V \times V$, i.e. E is a set of pairs of elements in V. The elements of V are the* vertices *of G – some authors also use the term* nodes *– and the elements of E are the* edges*,*

also called oriented edges *or* arcs[1]*, of G. An edge of the form (v, v) is a* ***loop*** *on v. Another way to express that E is a subset of $V \times V$ is to say that E is a* binary relation *over V. If either (u, v) or (v, u) belongs to E, the vertices u and v are* adjacent*. If neither (u, v) nor (v, u) belong to E, then u and v are* independent*. Given a digraph G, the set of vertices (respectively of edges) of G is denoted by $V(G)$ (respectively $E(G)$).*

The vast majority of the graphs that we will encounter are *finite* meaning that the set V of vertices is finite, and thus E contains at most $(\#V)^2$ edges.

REMARK 1.2.– *It is common to speak of the* order *of G for $\#(V(G))$ and the* size *of G for $\#(E(G))$.*

There are a few examples of infinite graphs in this book; see examples 1.47 and 4.11 (in formal language theory) and section 7.2 about colorings. Infinite graphs usually require more sophisticated arguments such as the axiom of choice. Implementation of infinite graphs in a computer could be tricky or impossible. From a practical point of view, particular instances of infinite graphs with a countable number of vertices and edges can be implemented. Think about a periodic graph that permits one to store only a finite amount of information to be repeated or a relation among vertices that can be computed and implemented as a function (see exercise 6 and example 1.5).

A digraph $G = (V, E)$ is said to be **simple** if E is a subset of $(V \times V) \setminus \{(v, v) \mid v \in V\}$. In that case, the relation E is *irreflexive*. Otherwise stated, loops are not allowed.

The elements belonging to a set are pairwise distinct: there is no repeated element. What we need to define a directed multigraph, i.e. a digraph where multiple edges between two vertices are allowed, is to permit repetitions of an element belonging to a set. In set theory, we can introduce the notion of a *multiset*. First, we restrict ourselves to multisets with finite integer multiplicities. A **multiset** M is a pair (S, m) where S is a set, in the "classical" sense, and $m : S \to \mathbb{N}_{\geq 1}$ is a *multiplicity function* that specifies the number $m(s)$ of occurrences of $s \in S$ in the multiset. As an example, the multiset denoted by $\{u, u, v, v, v, w\}$ is built from $S = \{u, v, w\}$ and

1 If we really have to distinguish the directed graphs from the undirected graphs that we will soon introduce, then we could restrict the use of the word "*edge*" to the undirected case and use the word "*arc*" solely in the directed case. But usually the context permits one to avoid any misunderstanding.

$m(u) = 2$, $m(v) = 3$, $m(w) = 1$. If the occurrences of an element have to be distinguished[2], we can index elements $s \in S$ by $s_1, \ldots, s_{m(s)}$. To continue the example, $\{u_1, u_2, v_1, v_2, v_3, w_1\}$ denotes the same multiset as above. If S is a finite set, then the cardinality of the multiset $M = (S, m)$ with finite multiplicities is

$$\#M := \sum_{s \in S} m(s).$$

Observe that a multiset (S, m) where $m(s) = 1$, for all $s \in S$, is simply a set. Equivalently, we could have defined the multiplicity function to map every element s of S to a finite subset of $\mathbb{N}$: the set of indices used for s.

Second, we could consider countable multiplicities[3]. In that case, an element of a multiset can be repeated infinitely many times and the multiplicity function maps every element to a subset of $\mathbb{N}$ (which is the set of indices used for that element). As an example, a vertex u could be repeated infinitely many times with prime indices: $\{u_2, u_3, u_5, u_7, u_{11}, \ldots\}$. Thus, the multiplicity function maps u to the set of prime numbers.

We now introduce a **directed multigraph** as a pair $G = (V, E)$ where V is a set of vertices and E is a multiset of edges built from a subset of $V \times V$. For a directed multigraph $G = (V, E)$, the fact that V is finite does not imply that E is also finite. Indeed, we could have infinitely many edges between two vertices. Thus, a directed multigraph is *finite* if both the set V and the multiset E are finite.

REMARK 1.3.– *It is common (and quite visual) to represent the vertices of a digraph by points in the plane (but we can also draw graphs on other surfaces like on a torus). Edges of the form (u, v) are represented by arrows going from u to v. We say that u (resp. v) is the **origin** (respectively, **destination**) of the edge. Actually any oriented arc of a curve can be used to join two vertices, not only straight vectors. Since positions of the vertices and arcs of curves joining the vertices can be freely chosen, there are usually infinitely many ways to represent a given graph. There is no reason that two edges that are intersecting in one representation are also intersecting in another representation of the same graph. We will rediscuss these notions with great care in section 6.1.*

2 For instance, to define a walk using different edges between two vertices.

3 Recall that a set is *countable* if it is in one-to-one correspondence with a subset of $\mathbb{N}$. Of course, from a mathematical point of view, further generalizations of multiplicity function and multisets can be considered; see section 1.7 for comments and pointers.

In Figure 1.1, we have depicted representations of a simple digraph, digraph and directed multigraph.

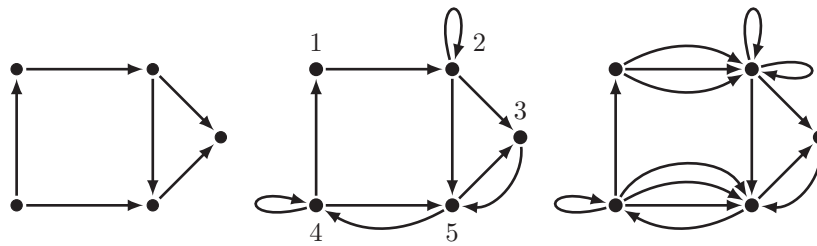

Figure 1.1. *From left to right: a simple digraph, a digraph and a directed multigraph*

A digraph G can be stored as an *adjacency list*: with each vertex u is associated the list of vertices v such that $(u, v) \in E(G)$. For the central digraph in Figure 1.1, the corresponding adjacency list is given in Table 1.1. A similar data structure can be used for directed multigraphs.

1	2		
2	2	3	5
3	5		
4	1	4	5
5	3	4	

Table 1.1. *An adjacency list*

EXAMPLE 1.4 (Tournament).– *A simple digraph $G = (V, E)$ where, for all pairs of distinct vertices u and v, either (u, v) or (v, u) belongs to E (but exactly one of these two edges belongs to E) is said to be a* tournament. *Indeed, it corresponds to an all-play-all tournament: each player plays against every other player and there are no ties. If u wins the confrontation against v, then we take the edge (v, u). See Figure 1.2.*

EXAMPLE 1.5.– *For an example of infinite simple digraph, take $\mathbb{N}_{>1}$ as set of vertices and a pair (m, n) of integers greater than 1 is an edge if and only if m divides n. The first few vertices and some edges of this digraph are depicted in Figure 1.3. Note that the relation E is* transitive. *If (m, n) and (n, p) belong to E, then (m, p) belongs to E.*

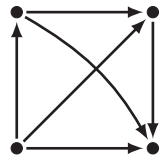

Figure 1.2. *A tournament among four players or teams*

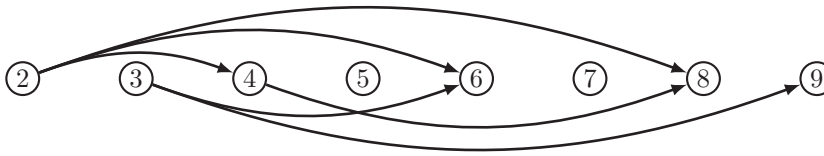

Figure 1.3. *A divisibility relation (first few vertices only)*

EXAMPLE 1.6.– *We consider the digraph made up of Webpages and there is an edge from a page p to a page q if there is a link on p referencing q. This digraph is finite but contains several billions of vertices. Independently of the content of the pages, here we are interested in the links that one user can follow by browsing through pages. Based on Perron's theorem (theorem 9.2), we will discuss the basis of the Google's PageRank algorithm in Chapter 10.*

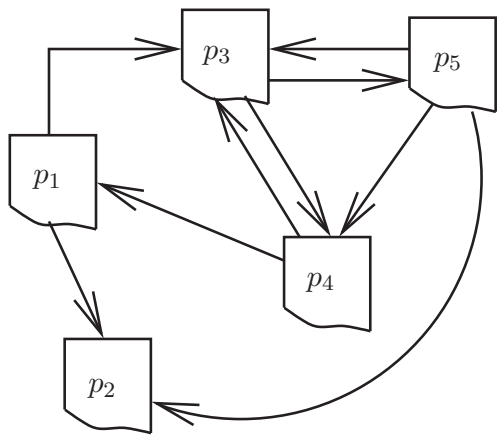

Figure 1.4. *Some links and Webpages*

Similarly to pages referencing other pages, we can also think about scientific papers that are citing other papers. In that case, we get a digraph where it is meaningful to try to identify relevant or influential papers. Which are the papers that are cited by many other papers, which are the "best" journals? The website http://www.eigenfactor.org/ uses a similar strategy to rank journals instead of Webpages [BER 07, WES 10].

EXAMPLE 1.7.– *The digraph that we may associate with Twitter is another example about social networks. There is an edge from a user account u to a user account v, if u is following the tweets of v. Therefore, all the tweets posted by v are displayed in the follower's timeline. Such a digraph captures who is following who. For instance, see [YAM 10] for an example of a User–Tweet digraph.*

REMARK 1.8.– *The reader may wonder about this triple definition of digraphs: simple digraphs, digraphs and directed multigraphs. Why should we take into account the case of simple or multiple digraphs? The answer is a pragmatic one. We choose the model that best fits the situation that we are considering. If we are interested in finding a shortest path between two vertices, it is meaningless to consider loops or multiple edges; going through a loop just makes the path longer. We just want to know if the two vertices are connected or not. In such a case, we will deal with simple digraphs. On the other hand, assume that we have to model the fact that between two cities, there is a road, a river but also a railway. Here multigraphs are useful to take this fact into account. Note that simple digraphs are special cases of digraphs that are themselves special cases of directed multigraphs. We reach same conclusions when we have to choose between digraphs and the unoriented graphs that we will soon introduce to model a particular situation.*

DEFINITION 1.9.– *In a directed graph $G = (V, E)$, we may associate two sets with every vertex v, respectively, the sets of* outgoing edges *and* incoming edges*:*

$$\omega^+(v) := \{(v, w) \in E \mid w \in V\}, \quad \omega^-(v) := \{(u, v) \in E \mid u \in V\}.$$

These definitions are extended to directed multigraphs and in that case, the corresponding sets are multisets. If, for all vertices v, the multisets $\omega^+(v)$ and $\omega^-(v)$ are finite, then we say that G is of *finite degree*. The **successors** (respectively, **predecessors**) of a vertex v are the vertices w (respectively, u) such that (v, w) (respectively, (u, v)) belongs to $\omega^+(v)$ (respectively, $\omega^-(v)$). The set of successors (respectively, predecessors) of v is denoted by $\mathsf{succ}(v)$ (respectively, $\mathsf{pred}(v)$). Note that there is a loop on v if and only if

$v \in \mathsf{succ}(v) \cap \mathsf{pred}(v)$. The **neighbors** of v are the vertices in $\mathsf{succ}(v) \cup \mathsf{pred}(v)$, they are the vertices adjacent to v. If there is no loop on v, then v is not a neighbor of itself. The set of neighbors of v, $\mathsf{N}(v) := \mathsf{succ}(v) \cup \mathsf{pred}(v)$, is sometimes called the *(open) neighborhood* of v. If v has to be included in the neighborhood, we speak of the *closed neighborhood* of v and is denoted by $\mathsf{N}[v]$.

In a directed multigraph of finite degree, the **indegree** of the vertex v is the number of incoming edges of v. It is denoted by $\mathsf{deg}^-(v) := \#\omega^-(v)$. The **outdegree** of the vertex v, denoted by $\mathsf{deg}^+(v) := \#\omega^+(v)$, is the number of outgoing edges of v. If $\mathsf{deg}^-(v) = 0$, v is a *source*. If $\mathsf{deg}^+(v) = 0$, v is a *sink*. If there exists k such that $\mathsf{deg}^+(v) = k$ for all vertices v, then the directed multigraph is said to be k**-regular**.

EXAMPLE 1.10.– *In theoretical computer science, a (complete)* deterministic finite automaton *is a k-regular directed multigraph where k is the size of the alphabet. Automata are, for instance, useful when working with regular expression and searching a word in a text. An example is given in Figure 1.5 where the alphabet is $\{R, B\}$ and the directed multigraph is* 2*-regular. See, for instance, [SUD 06] for a general reference.*

With infinite digraphs having infinitely many edges, indegrees or outdegrees may be infinite. For instance, the outdegree (respectively, indegree) of every vertex in the digraph of example 1.5 is infinite (respectively, finite). Indeed, every positive integer n is a divisor of all numbers of the form kn but every integer m has a finite number of divisors. Sources in this simple digraph are exactly the prime numbers.

The following observation is a direct consequence of the fact that every edge (in particular, loop) has exactly one origin and one destination.

LEMMA 1.11 (Handshaking Formula).– *Let $G = (V, E)$ be a finite directed multigraph. We have*

$$\sum_{v \in V} \mathsf{deg}^+(v) = \sum_{v \in V} \mathsf{deg}^-(v) = \#E.$$

DEFINITION 1.12 (Labeled Graphs).– *We can add some relevant information on the edges or vertices of a digraph. Formally, edges can receive a* label *or a* weight *(the latter term usually refers to numerical assignments). If $G = (V, E)$ is a directed multigraph, then we consider a map $\ell : E \to S$ where S is a set. For instance, S can be a finite set if we need to distinguish several types of edges (e.g. colors) or S could be equal to $\mathbb{N}$ or $\mathbb{R}$ if we need to add numerical*

information (e.g. cost, capacity and distance). Similarly, we can define a map of domain V to add information about the vertices.

EXAMPLE 1.13.– *Consider the 197 countries in the world and the flow of migrants moving from one country to another. So an edge from a country c to a country d will receive extra information to count the number of migrants from c to d during a given period. For instance, between 2005 and 2010, 1.8 million migrants moved from Mexico to the United States. Summing up the weights of the edges in-going to the United States give the total number of migrants entering the United States [ABE 14].*

EXAMPLE 1.14.– *On the Internet, Internet service providers represent their local network by a digraph where each edge is weighted by the capacity of the link.*

EXAMPLE 1.15.– *Consider the digraph depicted in Figure 1.5. Here, we have added labels B or R to edges corresponding to the two colors "Blue" or "Red". This is an example of a* synchronized digraph*: starting from any vertex, following a sequence of three consecutive edges of color blue, red and blue leads to vertex* 1*. This is a solution to the* road coloring problem *(See section 1.7).*

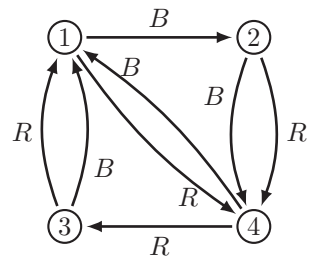

Figure 1.5. *Adding labels to edges*

DEFINITION 1.16.– *Let $G = (V, E)$ be a directed multigraph. Let W be a subset of V and F be a subset of*[4] *$E \cap (W \times W)$. The directed multigraph*

4 Here, the intersection of a multiset E and a set $W \times W$ has to be understood as follows: we keep the multiplicities carried by E but we restrict ourselves to edges with both endpoints in W. Now if we take a *subset* of $E \cap (W \times W)$, the multiplicity of an edge in this multiset is less than or equal to the corresponding multiplicity in E.

*$G' = (W, F)$ is a **subgraph** of G. We can also say that G is a* supergraph *of G'. A* proper subgraph *of G is a subgraph G' of G such that $G' \neq G$. The directed multigraph*

$$(W, E \cap (W \times W))$$

is the subgraph of G induced *by W. If e is an edge, $G - e$ denotes the subgraph where e has been deleted. If v is a vertex, $G - v$ denotes the subgraph of G induced by $V \setminus \{v\}$. These two operations can be extended to $G - F$ and $G - W$ where F is a multiset of edges and W is a set of vertices.*

*Now, we keep all the vertices of the original directed multigraph. A directed multigraph $H = (W, F)$ is a **spanning subgraph** of $G = (V, E)$ if $W = V$ and $F \subseteq E$. We also say that H is a* factor *of G. Otherwise stated, we only remove some of the edges of G. We will reconsider this notion in definition 4.5 with spanning subtrees. If a directed multigraph has m edges, then it has 2^m spanning subgraphs.*

1.1.2. *Unoriented graphs*

We now consider particular digraphs: the unoriented graphs.

DEFINITION 1.17.– *A multigraph $G = (V, E)$ is* unoriented *if, for every edge (u, v) belonging to E, the edge (v, u) belongs to E. Moreover, the edges (u, v) and (v, u) have the same multiplicities. Otherwise, stated E is symmetric. In that case, we allow ourselves to denote an edge between two distinct vertices by a set $\{u, v\}$, instead of taking the two pairs (u, v) and (v, u) into account. So the two oriented edges are identified*[5] *with a single (unoriented) edge $\{u, v\}$. This is just an abuse of notation. We say that u and v are* adjacent. *Similarly a loop on u can be designated by the multiset $\{u, u\}$. By abuse of notation, we may write $\{u, v\} \in E$ or $\{u, u\} \in E$. In a representation of an unoriented graph, we use segments or arcs of curves without any orientation. We can define accordingly* unoriented graphs *where multiple edges are not allowed and* simple unoriented graphs *where also loops are not allowed.*

REMARK 1.18.– *Even though it may sound awkward, note that an unoriented multigraph is a special case of a directed multigraph where*

5 When discussing walks it is important to note when there is an edge $\{u, v\}$, an agent can move from u to v and also from v to u. Thus in the unoriented case, we will not distinguish any orientation from u to v or from v to u.

orientation is neglected. Thus, definitions given for digraphs usually hold for unoriented graphs.

> ⚠ If not specified, the term *graph* (respectively, *multigraph*) is used only for unoriented graphs (respectively, multigraphs). From now on, we will use, respectively, the words graph and digraph to distinguish between the unoriented case and the general case.

EXAMPLE 1.19.– *Again from the world of social networks, we can view Facebook as a graph where the vertices are the Facebook users and there is an edge between two users whenever they are friends. "Being friends" is a symmetric relation, so there is no need to consider an orientation.*

EXAMPLE 1.20.– *A map of Belgium with the main cities connected by highways is an example of a simple graph where the edges have weights (distances in kilometers). If there is a highway from a to b, there is always a highway from b to a. One representation of the graph is depicted in Figure 1.6.*

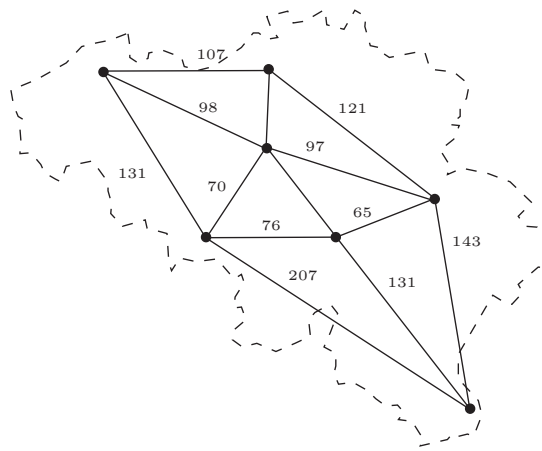

Figure 1.6. *Brugge, Antwerp, Brussels, Mons, Namur, Liège, Arlon (from north to south and from west to east)*

As a result of remark 1.18, the notions of degree, neighborhood or subgraph can be adapted to unoriented graphs. Let v be a vertex. We let

$\omega(v) := \{\{v, w\} \in E \mid w \in V\}$ be the set of edges adjacent to v. For a multigraph, $\omega(v)$ is usually a multiset. If $\omega(v)$ is finite, the **degree** of v, denoted by $\deg(v)$, is equal to the number of edges of the form $\{v, w\} \in E$ with $v \neq w$ plus twice the number of loops on v. Therefore, for a finite multigraph the handshaking formula becomes

$$\sum_{v \in V} \deg(v) = 2\,\#E. \qquad [1.1]$$

We transpose the notion of k-regularity to undirected graphs. Let k be an integer. The multigraph G is **k-regular**, if every vertex has degree k. As an example, the complete graph K_n introduced below is an $(n-1)$-regular graph. Regularity provides structural information about the graph.

The notion of k-regular graphs will be encountered several times in this book: about their spectrum with Hoffman theorem and algebra of matrices in section 8.5 and proposition 9.8, with PageRank in section 10.2. Similarly, complete graphs will be encountered in section 6.5, about coloring of planar graphs and Kuratowski's theorem (see proposition 6.13) or about Ramsey numbers in section 7.5.

EXAMPLE 1.21.– *The **complete graph** with n vertices is a simple graph where every edge between any two distinct vertices is present, i.e. every vertex is adjacent to all the other vertices. It is denoted by K_n. A* clique *in a graph G is a complete subgraph of G. The knowledge of the maximal cliques occurring in G provides structural information (for instance, think about a subgroup of people where everyone knows everyone else or a network where a region is densely connected) about G. Two examples are depicted in Figure 1.7.*

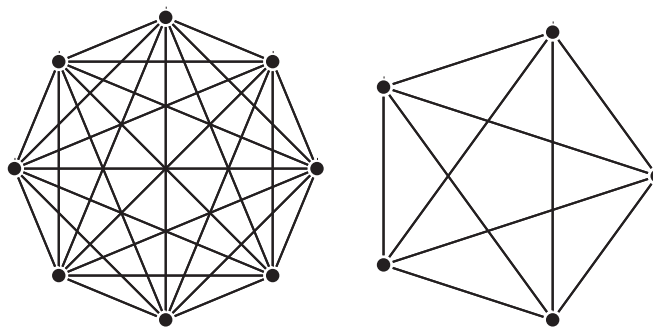

Figure 1.7. *The complete graphs K_8 and K_5*

REMARK 1.22.– *Every simple graph with n vertices is a (spanning) subgraph of K_n.*[6]

There are several extra notions that we usually encounter in an unoriented context (of course, it would not be hard to adapt them in the general case of digraphs). Roughly speaking, a vertex cover is about a subset of vertices that meets every edge of the graph and a dominating set is a subset of vertices "close" to every vertex.

DEFINITION 1.23 (Covering and domination).– *Let G be a graph. Let W be a subset of $V(G)$. If, for every edge $e \in E(G)$, there exists a vertex $v \in W$ such that v is an endpoint of e, then W is a* vertex cover *of G. A* minimum vertex cover *is a vertex cover of G of minimal size. A vertex cover of the Petersen graph*[7] *is represented in Figure 1.8 (left). It is a minimum vertex cover (see section 1.8).*

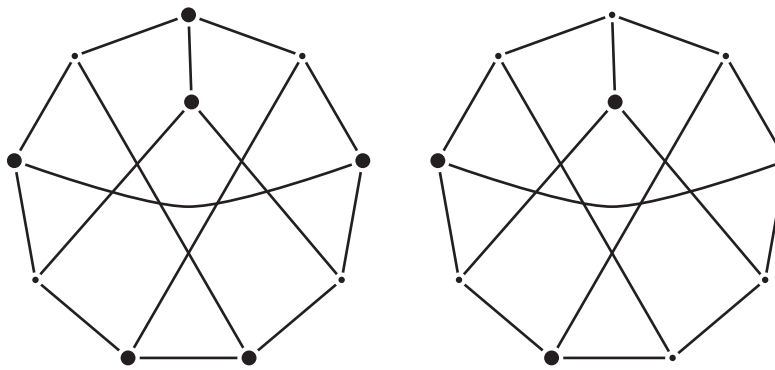

Figure 1.8. *A vertex cover (left) and dominating set (right) of the Petersen graph represented by large vertices*

With the same philosophy, a subset W of $V(G)$ is a dominating set *of G, if every vertex of $V(G)$ is either in W or adjacent to a vertex in W, i.e. $V(G) =$*

6 In a second reading, make also the connection with lemma 7.29.

7 The *Petersen graph* is recurrent in graph theory. It is common to find it as example or counterexample for many problems. Petersen built this graph as the smallest bridgeless 3-regular graph with no 3-edge-coloring [PET 98]. (These notions will be defined later on in this book).

$(\cup_{w\in W}\mathsf{N}(w))\cup W=\cup_{w\in W}\mathsf{N}[w]$ *where we recall that* $\mathsf{N}(w)$ *(respectively,* $\mathsf{N}[w]$*) is the neighborhood of* w *(respectively, closed neighborhood of* w*). A dominating set of the Petersen graph is represented in Figure 1.8 (right). Note that this set of three vertices is not a vertex cover: at least five edges have no endpoints in this set.*

DEFINITION 1.24.– *A simple graph* $G=(V,E)$ *is* bipartite *if there exists a partition of* V *into two subsets* V_1 *and* V_2 *in such a way that every edge in* E *is of the form* $\{v_1,v_2\}$ *with* $v_1\in V_1$ *and* $v_2\in V_2$*. The pair* (V_1,V_2) *is said to be a* ***bipartition*** *of* G*. The* complete bipartite graph $K_{\ell,m}$ *is a bipartite graph with* $\ell+m$ *vertices such that there is a bipartition* (V_1,V_2) *with* $\#V_1=\ell$ *and* $\#V_2=m$ *and for every* $u\in V_1$ *and every* $v\in V_2$*, the edge* $\{u,v\}$ *is present. A complete bipartite graph of the form* $K_{1,m}$ *is said to be a* star.

This notion of partition can easily be generalized. Let $n\geq 2$*. A simple graph* $G=(V,E)$ *is* n-partite *if there exists a partition of* V *into* n *subsets* $V_1,\ldots,V_n$ *in such a way that every edge in* E *is of the form* $\{u,v\}$ *with* $u\in V_i$ *and* $v\in V_j$ *for some* i,j *such that* $i\neq j$*. The* complete n-partite graph $K_{m_1,\ldots,m_n}$ *has a set of vertices partitioned into* n *subsets* $V_1,\ldots,V_n$ *in such a way that* $\#V_i=m_i$*; for all* i*, there is no edge between two vertices of the same subset* V_i *and for all vertices* u *in* V_i *and* v *in* V_j*,* $i\neq j$*, the edge* $\{u,v\}$ *is present.*

Compared with Figure 1.9, the complete tripartite graph $K_{3,3,2}$ has 21 edges.

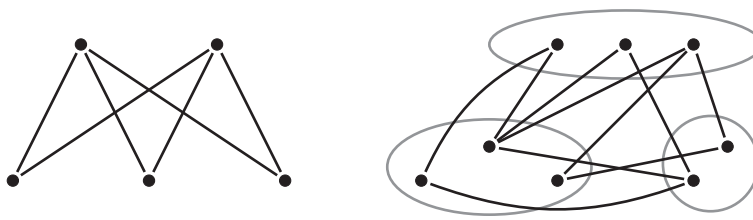

Figure 1.9. *The complete bipartite graph* $K_{2,3}$ *and a* 3*-partite graph*

EXAMPLE 1.25.– *Consider a graph where the vertices represent either workers, or tasks to be completed. So there is a natural partition of this set. There is an edge between a worker* W_i *and a task* T_j *if* W_i *has the skills to perform* T_j*. Usually the question is to assign tasks to workers in such a way that every task will be realized. In Figure 1.10, a spanning subgraph of the bipartite graph is represented with black edges.*

For a modeling of the Internet topology by bipartite graphs, see [TAR 13].

EXAMPLE 1.26.– *Another example of modeling is to consider patients needing kidney transplants and donors. With blood or tissue incompatibility, some matchings are impossible. A bipartite graph can model the possible compatible patient–donor pairs. We can consider a two-way kidney exchange that involves two patients, each of whom is incompatible with his/her own donor but compatible with the other donor (see, for instance*[8]*, [ROT 07]).*

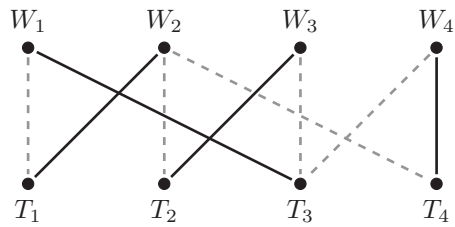

Figure 1.10. *Workers and tasks, in black a spanning subgraph*

1.2. Paths and connected components

Let us place an agent on a vertex of a graph. This agent is allowed to move from vertex to vertex along the edges of the graph. This leads to the notion of a walk. If the agent moves forever, even though the graph is finite, this will gives infinite paths (the same vertex can be visited several times). For a general presentation, we consider the directed case. We refer again to remark 1.18.

DEFINITION 1.27.– *Let $G = (V, E)$ be a directed multigraph. A* ***walk*** *in G is a finite or infinite sequence of edges $((v_{i,1}, v_{i,2}))_i$ such that $v_{i,2} = v_{i+1,1}$ for all i. The* length *of a finite walk is the number of edges in the sequence. To a walk $((v_{i,1}, v_{i,2}))_{i=1,\dots,n}$ of length n corresponds the sequence of the $n+1$ visited vertices $(v_{1,1}, v_{2,1}, \dots, v_{n,1}, v_{n,2}) = (v_{1,1}, v_{1,2}, v_{2,2}, \dots, v_{n,2})$. This is a walk from (or joining) $v_{1,1}$ to $v_{n,2}$.*

The walk is ***closed*** *if $v_{1,1} = v_{n,2}$. A closed walk is defined up to a cyclic permutation of the edges in the sequence. In a digraph (where there are no multiple edges), a walk is completely determined by the sequence of visited vertices.*

8 Alvin Elliot Roth won the Nobel prize in economic sciences in 2012.

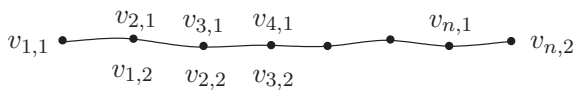

Figure 1.11. *A walk*

*A **trail** is a walk where all the edges occurring in the sequence are pairwise distinct. Finally, a **path** (some authors use the term* simple path *to emphasize the special case) is a walk where all the visited vertices (except maybe the first one and the last one when defining a closed path) are pairwise distinct. This implies that every path is a trail. A closed path is usually called a **cycle**. A graph without cycle is said to be* acyclic.

For the digraph in Figure 1.12, $(e_1, e_4, e_5, e_4, e_6)$ is a walk joining 1 to 5; $(e_1, e_4, e_5, e_2, e_3)$ is a closed trail (no repeated edges but the vertex 2 is visited twice); (e_1, e_4, e_6, e_7) is a path joining 1 to 6. There are exactly three cycles: (e_1, e_2, e_3), (e_4, e_5) and (e_6, e_7, e_8) and three closed trails which are not cycles: $(e_1, e_4, e_5, e_2, e_3)$, $(e_4, e_6, e_7, e_8, e_5)$ and $(e_1, e_4, e_6, e_7, e_8, e_5, e_2, e_3)$. For the simple graph on the right, $(f_1, f_4, f_5, f_6, f_7, f_4, f_2)$ is a walk joining a and c.

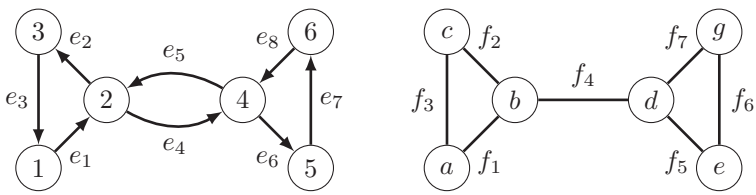

Figure 1.12. *A simple digraph and a simple graph*

⚠ Here is a major difference with the oriented case: There is no trail joining a to c and passing through d because, for any such walk, the (unoriented) edge f_4 must be repeated. In particular, in an unoriented graph, there is no cycle of length exactly 2. To have a such a cycle, we need two edges between two given vertices, i.e. a multigraph. Thus, there are exactly two cycles: (f_1, f_2, f_3) and (f_5, f_6, f_7).

EXAMPLE 1.28 (Collaboration Graphs).– *In science, a majority of published research papers are coauthored by several researchers. We may build a graph where the vertices are the scientists and two scientists are connected if they share a publication. For instance, the so-called* Erdős number *of a mathematician M is the distance in this graph, i.e. the length of the shortest path (assuming that it exists), between M and the famous Hungarian mathematician Paul Erdős*[9]*. See the paper [ODD 79] written by R. Graham (using a pseudonym) or [EAS 10, p. 34]. It seems that every living mathematician has a quite small Erdős number (less than seven). This is not a theorem, it just reflects the fact that the community of researchers is quite dense. This "fact" is known as the* small world phenomenon. *See the comments in section 1.7.*

If you are a tennis player, you can try to compute your own Federer number: players who played once (winning or losing) an official game against Roger Federer have a Federer number equal to 1*, players who played against a player with a Federer number* 1 *but who never played against Federer himself, have a number equal to* 2 *and so on.*

EXAMPLE 1.29.– *On various operating systems, you can find the tool* `traceroute` *that reveals the presence of intermediate-level equipment (routers) on the route (or path) of packets of data taken from an IP network on their way to a given host on the Internet.*

1.2.1. *Connected components*

Let $G = (V, E)$ be a directed multigraph. Let u, v be two vertices. We say that u is *connected* to v and we write $u \to v$, if there exists a walk from u to v. In particular, we assume that every vertex u is connected to itself (with a length-0 walk). If u is connected to v, we let $\mathsf{d}(u, v)$ denote the minimal length of a walk from u to v. Note that there is always a path achieving $\mathsf{d}(u, v)$. A *strongly connected component* (**SCC**) is a maximal subset C of V such that for all $u, v \in C$, $u \to v$ (in particular, we also have $v \to u$). The term "strongly" reflects the fact that orientation is taken into account. If $V(G)$ is an SCC, then G is said to be *(strongly) connected*.

9 Paul Erdős (1913–1996) has had more than 500 collaborators, he worked in several fields ranging from combinatorics and graph theory to number theory, analysis and probability. For instance, see [HOF 99] for an account about this colorful mathematician.

An SCC is *trivial* if it is restricted to a single vertex with no loop. In every non-trivial SCC, there exists a closed walk visiting all the vertices of the component. We write $u \leftrightarrow v$ if and only if $u \to v$ and $v \to u$. Since every vertex is connected to itself, we have $u \leftrightarrow u$ for all vertices u. Note that $\leftrightarrow$ defines an equivalence relation over $V(G)$ and the corresponding equivalence classes are exactly the SCCs of G.

Since (unoriented) multigraphs are special cases of directed multigraphs (see remark 1.18), we have thus defined connected vertices and connected components in a multigraph.

A directed multigraph $G = (V, E)$ is *weakly connected* if the unoriented graph obtained by taking the symmetric closure of E is connected. That is to say, if one disregards the orientation of the oriented edges, then the graph is connected.

EXAMPLE 1.30.– *The digraph depicted in Figure 1.13 is not strongly connected but weakly connected. We have* $1 \to 9$ *but* $9 \not\to 1$ *or simply,* $2 \to 5$ *but* $5 \not\to 2$. *It has four SCCs:* $\{1, 2, 3, 4\}$, $\{5\}$, $\{6, 7\}$ *and* $\{8, 9\}$.

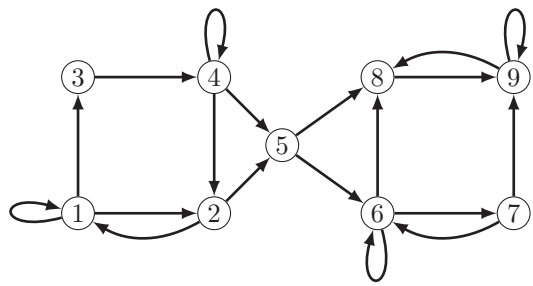

Figure 1.13. *A digraph and its four SCCs*

REMARK 1.31 (Distance).– *The fact that* u *is connected to* v *does not imply, in a digraph, that* v *is also connected to* u. *The relation "being connected" over* V *is not necessarily symmetric. But for an unoriented multigraph* G, *since* $E(G)$ *is symmetric, so is the relation "being connected". In that case, the relations* $\to$ *and* $\leftrightarrow$ *coincide and the map* d *restricted to a connected*

component is a distance[10]*. Note that we usually still refer to the term* distance *in a strongly connected digraph even if the map* d *is not symmetric. In Figure 1.13, we have* $\mathsf{d}(1,3) = 1$ *but* $\mathsf{d}(3,1) = 3$.

The procedure given in Table 1.2 allows us to compute the reflexive and transitive closure of $\mathsf{succ}(v)$, i.e. the set $\mathsf{succ}^*(v) := \{u \in V \mid v \to u\}$. Note that since $v \to v$, v belongs to $\mathsf{succ}^*(v)$. In the following algorithm, the data are a finite digraph $G = (V, E)$ and a vertex $v \in V$, the output is the set $\mathsf{succ}^*(v)$. The idea is to let the set $Component$ grow by adding elements in $\mathsf{succ}(u)$ for the vertices u that have been recently added to $Component$ and stored in New. When no new vertices are added, the procedure stops.

TRANSITIVECLOSURESUCC(G, v)
1 $Component \leftarrow \{v\}$;
2 $New \leftarrow \{v\}$;
3 **while** $New \neq \emptyset$,
4 **do** $Neighbors \leftarrow \emptyset$;
5 **for** all $u \in New$,
6 **do** $Neighbors \leftarrow Neighbors \cup \mathsf{succ}(u)$;
7 $New \leftarrow Neighbors \setminus Component$;
8 $Component \leftarrow Component \cup New$;
9 **return** $Component$;

Table 1.2. *Algorithm to compute* $\mathsf{succ}^*(v)$

Similarly, we compute $\mathsf{pred}^*(v) := \{u \in V \mid u \to v\}$. The SCC[11] of u is simply $\mathsf{succ}^*(u) \cap \mathsf{pred}^*(u)$. The procedure can be adapted to detect connected components of an unoriented graph. In particular, a graph is connected if and only if $V = Component$. Also see the Roy–Warshall algorithm in section 1.8.

1.2.2. *Stronger notions of connectivity*

To conclude this section, we mention stronger notions related to connectedness. All notions and results are stated in an unoriented setting.

10 In the mathematical sense, $\forall u, v$, $\mathsf{d}(u,v) \geq 0$ and $\mathsf{d}(u,v) = 0$ if and only if $u = v$, $\mathsf{d}(u,v) = \mathsf{d}(v,u)$ and d satisfies the triangular inequality: $\forall u, v, w$, $\mathsf{d}(u,v) \leq \mathsf{d}(u,w) + \mathsf{d}(w,v)$.

11 For more insight, consider *Tarjan's SCCs algorithm* [TAR 72].

DEFINITION 1.32.– *In a multigraph, a* ***bridge*** *(also called* isthmus *or* cut-edge*) is an edge whose removal increases the number of connected components in the graph. In particular, if G is a connected multigraph, a bridge is an edge e such that $G - e$ is not connected. Observe that in that case, $G - e$ has exactly two connected components. Since E is connected, every vertex in $G - e$ is connected to an endpoint of e.*

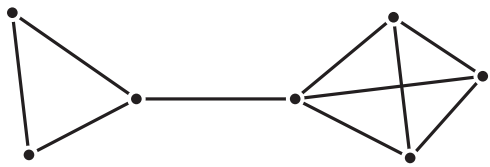

Figure 1.14. *A graph with a single bridge*

Clearly, an edge is a bridge if and only if it does not belong to any cycle. Equivalently, e is a bridge if and only if there exist two connected vertices u and v such that all paths joining u and v go through e. As an example, the edge f_4 in Figure 1.12 is a bridge. In a directed multigraph, a *strong bridge* is an edge such that removing that edge increases the number of SCC in the digraph.

THEOREM 1.33 (Robbin's theorem — strong orientation).– *Let G be a simple graph. There exists an orientation for every edge, turning it into a strongly connected digraph, if and only if G is connected and has no bridge.*

The proof is left as an exercise: it is clear that if G has a bridge, then no orientation may exist. Notice that this result is still valid for a multigraph. Indeed, if there are at least two edges between two vertices u and v, locally the task is easy: one simply takes the two orientations (u, v) and (v, u) (see [ROB 39]).

The notion of a bridge extends to a *cut-set*.

DEFINITION 1.34 (Cut-set).– *Let G be a multigraph. A subset F of $E(G)$ is a cut-set if $G - F$ has more connected components than G. In particular, when G is a connected multigraph, a set F such that $G - F$ is disconnected is a cut-set.*

DEFINITION 1.35.– *Let us introduce a quantity $\lambda(G)$ (some authors use $\kappa'(G)$). If G is a disconnected multigraph, we set $\lambda(G) = 0$. If G is a*

*connected multigraph, $\lambda(G)$ is defined as the cardinality of a cut-set of minimal size. Let $k \geq 1$. We say that a multigraph G is k-**edge connected** when*

$$\lambda(G) \geq k.$$

This means that G is connected and removing any subset of at most $k-1$ edges leaves the graph connected. Note that if $\lambda(G) = k$, then removing k well-chosen edges leads to a disconnected graph (with exactly two connected components).

For instance, $\lambda(K_n) = n - 1$ and a connected graph with no bridge is 2-*edge connected*: we have to delete at least two edges to disconnect the graph. See, for instance, the graph depicted in Figure 5.1.

Dual notions (edges versus vertices) can be defined when removing some vertices of the graph (and, of course, the edges adjacent to them).

DEFINITION 1.36.– *A **cut-vertex** is a vertex such that its removal increases the number of connected components in the multigraph. A subset W of $V(G)$ is a **separating set** if $G - W$ has more connected components than G. In particular, if G is connected, a vertex v (respectively, a set W) such that $G - v$ (respectively, $G - W$) is disconnected, is a cut-vertex (respectively, a separating set).*

REMARK 1.37.– *When removing a bridge, the number of connected components increases by 1. The situation is quite different when removing a cut-vertex. Removing the central vertex in Figure 1.15 leads to three connected components.*

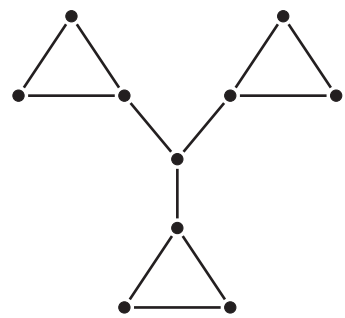

Figure 1.15. *A cut-vertex*

Every connected non-complete multigraph has a separating set. It is thus legitimate to introduce the next notion.

DEFINITION 1.38 (Vertex Connectivity).– *Let G be a multigraph. We define a quantity $\kappa(G)$ as follows. We set $\kappa(G) = 0$ whenever G is disconnected. For a complete graph K_n, we set $\kappa(K_n) = n - 1$ because it cannot be disconnected when removing vertices. In all other cases, G is a connected and non-complete graph, $\kappa(G)$ is the cardinality of a separating set of minimal size.*

*Let $k \geq 1$. We say that a multigraph G is k-**connected** (more precisely,* k-vertex connected*) if*

$$\kappa(G) \geq k.$$

This means that G is connected and removing any subset of at most $k-1$ vertices leaves the graph connected[12]. The least integer k such that G is k-connected is the vertex connectivity *of G. If $\kappa(G) = k$, then removing k well-chosen vertices leads to a disconnected graph or to a trivial graph reduced to a single vertex.*

For instance, every cycle of length at least 3 is 2-connected. The next result gives an interpretation of the vertex-connectivity.

THEOREM 1.39 (Menger's theorem).– *Let $G = (V, E)$ be a finite graph. Let u, v be two non-adjacent vertices. Let $k \geq 0$ be the smallest integer such that there exists $W \subset V$ with $\#W = k$ and $u \not\leftrightarrow v$ in the subgraph $G - W$. The maximal number of vertex-independent paths joining u and v, i.e. any two such paths have no common vertex except for u and v, is equal to k.*

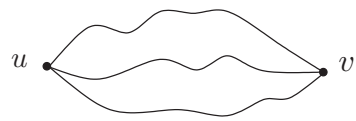

Figure 1.16. *Three vertex-independent paths*

A proof can be found in [DIR 66]. For extension to infinite graphs, see [HAL 74]. There is also an analogous version of this result in terms of edge connectivity and edge-independent paths.

12 It could be left with a single vertex but recall that a trivial graph reduced to a vertex is connected.

COROLLARY 1.40.– *Let $k \geq 2$. A graph is k-connected if and only if every pair of distinct vertices is connected by at least k vertex-independent paths.*

To conclude this section, we mention the following result.

THEOREM 1.41 (Whitney [WHI 32]).– *For every graph, we have*

$$\kappa(G) \leq \lambda(G) \leq \min_{v \in V(G)} \deg(v).$$

PROOF.– The right inequality is clear: if we remove all the edges adjacent to a vertex of minimal degree, then this vertex will be disconnected from the other vertices.

If $\lambda(G) \leq 1$, then $\kappa(G) = \lambda(G)$. Now assume that $\lambda(G) = k \geq 2$. There exist k edges $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$ whose removal leads to a disconnect graph with a partition of the vertices into two sets V_1 and V_2 corresponding to the two connected components of the resulting graph[13]. Note that the u_i's and v_i's are not necessarily different. We may assume that the u_i's belong to V_1 and the v_i's to V_2. If there exists $w \in V_1 \setminus \{u_1, \ldots, u_k\}$, then removing $u_1, \ldots, u_k$ will disconnect w from the vertices in V_2. So we remove at most (some of the u_i's could be identical) k vertices to disconnect G. The other case is when $V_1 = \{u_1, \ldots, u_k\}$. The argument is that u_1 has at most k neighbors, i.e. $\deg(u_1) \leq k$. Indeed, u_1 has at most $(\#V_1) - 1$ neighbors in V_1 and at most $k - ((\#V_1) - 1)$ in V_2 because we know that there are exactly k edges between V_1 and V_2 and every vertex in V_1 is the endpoint of at least one such edge. Since $\lambda(G) = k$ and from the right inequality of the statement, we deduce that $\deg(u_1) \geq k$. Hence, $\deg(u_1) = k$. Thus, we may remove the k neighbors of u_1 to disconnect the graph, meaning that $\kappa(G) \leq k$. ■

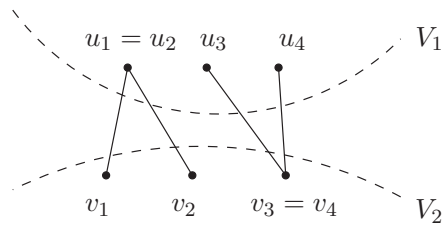

Figure 1.17. *Illustration of Whitney's theorem*

13 Removing $\{u_1, v_1\}, \ldots, \{u_{k-1}, v_{k-1}\}$ leaves the graph connected and $\{u_k, v_k\}$ becomes a bridge whose removal gives two connected components.

1.3. Eulerian graphs

In a directed multigraph, an **Eulerian trail** is a trail visiting all the edges, i.e. a walk going exactly once through all edges.

A directed multigraph is **Eulerian** if it has a closed Eulerian trail, i.e. a walk starting and ending in the same vertex and going exactly once through all edges. Such a closed Eulerian trail is usually said to be an *Eulerian circuit*. The fact that a digraph is or is not Eulerian is easy to check and the constructive proof below gives an algorithm for finding an Eulerian circuit.

LEMMA 1.42.– *A weakly connected (finite) directed multigraph $G = (V, E)$ is Eulerian if and only if, for all vertices $v \in V$, $\deg^-(v) = \deg^+(v)$.*

PROOF.– The proof is elementary, but it provides an algorithm to get a Eulerian circuit. Start from any vertex v_0. Choose an edge starting in this vertex. Repeat the procedure from the reached vertex: choose an edge among the edges that are still available (i.e. not yet chosen during a previous step). Since the graph is finite and since, for all vertices $v \in V$, $\deg^-(v) = \deg^+(v)$, after a finite number of choices, we are back to v_0. If the set of edges that have been already chosen is equal to E, we have obtained an Eulerian circuit. Otherwise, we extend the closed trail as follows. Pick in that trail a vertex v_1 such that there exists an edge with origin v_1 among the set of unchosen edges (such an edge exists because the graph is connected). Repeat the procedure from v_1 and get a new closed trail going through v_1 and merge in an appropriate way this closed trail with the first one to get a longer closed trail: start the trail from v_0, when reaching v_1 complete the second closed trail coming back to v_1, then finish the initial trail leading back to v_0. Repeat the procedure if there are edges left. The algorithm terminates because of the finiteness of the graph. ■

COROLLARY 1.43.– *A weakly connected (finite) directed multigraph $G = (V, E)$ has a Eulerian trail from u to v if and only if, $\deg^-(u) + 1 = \deg^+(u)$, $\deg^-(v) = \deg^+(v) + 1$ and, for all vertices $w \in V \setminus \{u, v\}$, $\deg^-(w) = \deg^+(w)$.*

PROOF.– If we add an edge (v, u) to the directed multigraph, then we are back to the previous lemma. ■

We can directly reformulate the result in the unoriented case.

COROLLARY 1.44.– *A connected (finite) multigraph $G = (V, E)$ is Eulerian if and only if all vertices have even degree. A connected (finite) multigraph*

$G = (V, E)$ has an Eulerian trail from u to v if and only if u and v are of odd degree and all other vertices have even degree.

This result gives the solution to the historical problem solved by Euler about the seven bridges of Königsberg: seven bridges were situated across the Pregolya River and the people living in Königsberg (now Kaliningrad) wanted to go for a walk on these bridges; however, their stroll should not use the same bridge twice. In other words, they were looking for what we call a Eulerian circuit. They were unsuccessful but Euler showed that, indeed, there is no such circuit (see Figure 1.18).

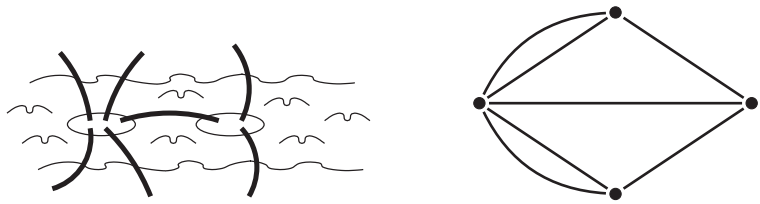

Figure 1.18. *The seven bridges, the borders of the river and two isles are represented by the four vertices*

Fleury's algorithm given in Table 1.3 connects two notions encountered so far: Eulerian graphs and bridges. But it has a drawback. To implement this method, we have to detect bridges efficiently[14]. Graph traversal and the detection of bridges leads to an algorithm whose running time is quadratic in $\#E$. The idea is to postpone the crossing of a bridge as much as possible. Indeed, when we cross a bridge, there is no way to go back to the component we were in without taking it a second time, which is not permitted. Thus that component must have been completely visited first (see the comments in section 1.7). Recall that $\omega(v)$ is the set of edges adjacent to v. Note in line 6 that the graph is updated. So we are looking for bridges in the subgraph restricted unvisited edges.

The output of Fleury's algorithm is a sequence of edges. Either the length of this sequence is equal to $\#E$ and we have found a Eulerian circuit or a Eulerian trail, or if we have less than $\#E$ edges in the sequence, then the graph is not Eulerian. To find an algorithm with a better performance, search for Hierholzer's algorithm on the web. Applying Fleury's algorithm to the graph

14 For instance, see Tarjan's Bridge-finding algorithm [TAR 74].

depicted in Figure 1.19, one possible output is represented (the edges have been ordered with respect to the output sequence of the algorithm). Note that after selecting the first three edges, the edge numbered 8 is a bridge of the remaining graph. Thus, we cannot choose that one at this step of the algorithm. Nevertheless, we could have chosen either edge 4 or edge 7 to pursue. The proof of the exactness of Fleury's algorithm is left as an exercise.

FLEURY(G, v_0) where $G = (V, E)$ is a multigraph and $v_0 \in V$

1 $i \leftarrow 1$;
2 **repeat**
3 **if** $\omega(v_{i-1})$ contains an edge that is not a bridge
4 **then** pick such an edge $e_i = \{v_{i-1}, v_i\} \in E$;
5 **else** pick a bridge $e_i = \{v_{i-1}, v_i\} \in E$;
6 $G \leftarrow G - e_i$; $i \leftarrow i + 1$;
7 **until** $i > \#E$
8 **return** $(e_1, e_2, \ldots)$;

Table 1.3. *Fleury's algorithm [FLE 83]*

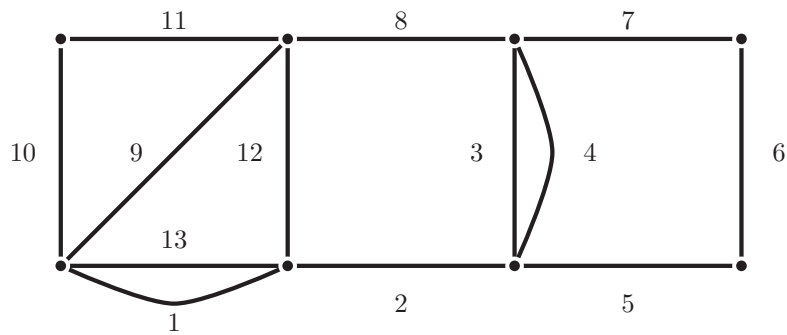

Figure 1.19. *An application of Fleury's algorithm (starting with lower-left vertex)*

1.4. Defining Hamiltonian graphs

A notion dual to Eulerian graphs (vertices versus edges) is the following one. In a digraph, a path is *Hamiltonian* if it visits all the vertices. In

Figure 1.13, the path visiting the vertices $2, 1, 3, 4, 5, 6, 7, 9, 8$ is the unique Hamiltonian path for this graph.

As in the Eulerian case where we have first defined an Eulerian trail, then a Eulerian graph, a digraph is **Hamiltonian** if there exists a cycle starting and ending in the same vertex and going exactly once through all the vertices. This cycle is a *Hamiltonian circuit*. If we can answer the question of whether or not a digraph is Hamiltonian, then we can also trivially answer the question if we allow multiple edges. So we can consider simple digraphs. Analogous definitions can be given in the unoriented case.

EXAMPLE 1.45.– *A trivial example is given by K_n, $n \geq 3$, where every permutation of the n vertices gives a Hamiltonian circuit. Hence, K_n has $n!$ distinct Hamiltonian circuits. Another example is depicted in Figure 1.20 where we have to find a circuit for a Knight on a chessboard in such a way that every square is visited once. So here, we have an underlying graph with 64 vertices and there is an edge between two vertices if there is a legal Knight's move between these two squares.*

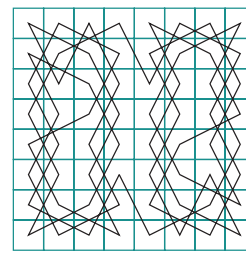

Figure 1.20. *A Knight's tour on a chessboard*

It turns out that finding a Hamiltonian circuit seems to be much more difficult than the Eulerian counterpart (lemma 1.42 and corollary 1.44). Indeed, deciding (using a generic algorithm) whether or not a graph is Hamiltonian is well known to be an NP-complete problem [GAR 79]. In Chapter 2, we make precise the latter notion. Chapter 3 will present necessary or sufficient conditions for a graph to be Hamiltonian.

1.5. Distance and shortest path

In this section, in great detail we present Dijkstra's algorithm computing one shortest path[15] from a vertex v_1 (single source) to every other vertex in the graph. We will consider simple weighted digraphs. Indeed, if several edges are connecting two vertices, we can simply consider the one of smallest weight. We can also disregard loops that will increase the total weight. Note that the ideas developed for Dijkstra's algorithm are similar to those found in Prim's algorithm for minimum spanning trees (see remark 4.8).

Let $G = (V, E)$ be a simple (finite) digraph and $\mathsf{w} : E \to \mathbb{R}_{\geq 0}$ be a weight function. If there is no weight function attached to G, we may assume that every edge has a weight equal to one (thus we will count the length of the corresponding walk and this notion is compatible with the distance discussed in remark 1.31). If $(e_1, \ldots, e_k)$ is a walk joining u and v, then the weight of this walk is

$$\sum_{j=1}^{k} \mathsf{w}(e_j)\,.$$

If we are interested in walks of minimal weight, we can restrict ourselves to paths from u to v. We also extend w to $V \times V$ with values in $\mathbb{R}_{\geq 0} \cup \{+\infty\}$ by setting $\mathsf{w}(u, v) = +\infty$ when $(u, v) \notin E$. As usual, $r + \infty = +\infty$ for all real numbers r. If ℓ is a list and x is an element, $\texttt{concat}(\ell, x)$ is the list obtained by appending x to ℓ.

A set X is initialized with $\{v_1\}$ and the idea is roughly to let this set grow until it is equal to V. At each step, one vertex is added to X. We choose the "best" candidate (see line 8). Then, we decide if we gain something for the remaining vertices using this new vertex (lines 10–13). It is remarkable that local decisions lead to a global solution.

EXAMPLE 1.46.– *To grasp the idea behind Dijkstra's algorithm given in Table 1.4, we first run it on a small example. Consider the weighted graph depicted in Figure 1.21. The first row in Table 1.5 corresponds to the initialization of the variables in lines 1–6 of the algorithm. For every vertex u, the variable $T(u)$ stores the value of the smallest path found so far and $C(u)$ is a sequence of vertices starting with v_1 realizing such a path from v_1 to u. In*

15 We write "one" shortest path and not "the" shortest path because several paths with minimal weight may exist.

line 8, we choose a vertex v having a minimal T value among the unchosen vertices. Then in lines 10–13, we update the variable T and C for the remaining unchosen vertices by determining if there is a benefit (line 11) using the vertex v.

DIJKSTRA(G, w, v_1) where $G = (V, E)$ is a simple digraph,
w a weight function and $v_1 \in V$

```
1   for all v ∈ V \ {v_1},
2       do T(v) ← w(v_1, v);
3          if T(v) ≠ +∞
4             then C(v) ← (v_1, v)
5             else  C(v) ← ();
6   X ← {v_1};
7   while X ≠ V
8       do pick v ∈ V \ X s.t. ∀y ∈ V \ X, T(v) ≤ T(y);
9          X ← X ∪ {v};
10         for all y ∈ V \ X,
11             do if T(y) > T(v) + w(v, y)
12                   then T(y) ← T(v) + w(v, y);
13                        C(y) ← concat(C(v), y);
```

Table 1.4. *Dijkstra's algorithm*

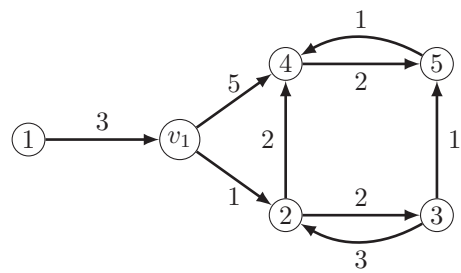

Figure 1.21. *A weighted (simple) digraph*

We now give a proof of the exactness of the algorithm. It is clear that the algorithm terminates when starting with a finite graph: on line 5, we consider each time a new vertex and the algorithm stops when all the vertices have been considered. We essentially follow the lines of [GIB 85] for the proof.

nth iteration	X	$T(1)$ $C(1)$	$T(2)$ $C(2)$	$T(3)$ $C(3)$	$T(4)$ $C(4)$	$T(5)$ $C(5)$
1	$\{v_1\}$	$+\infty$ $()$	$\mathbf{1}$ $(v_1,2)$	$+\infty$ $()$	5 $(v_1,4)$	$+\infty$ $()$
2	$\{v_1,2\}$	$+\infty$ $()$	1 $(v_1,2)$	3 $(v_1,2,3)$	$\mathbf{3}$ $(v_1,2,4)$	$+\infty$ $()$
3	$\{v_1,2,4\}$	$+\infty$ $()$	1 $(v_1,2)$	$\mathbf{3}$ $(v_1,2,3)$	3 $(v_1,2,4)$	5 $(v_1,2,4,5)$
4	$\{v_1,2,4,3\}$	$+\infty$ $()$	1 $(v_1,2)$	3 $(v_1,2,3)$	3 $(v_1,2,4)$	$\mathbf{4}$ $(v_1,2,3,5)$
5	$\{v_1,2,4,3,5\}$	$+\infty$ $()$	1 $(v_1,2)$	3 $(v_1,2,3)$	3 $(v_1,2,4)$	4 $(v_1,2,3,5)$
6	$\{v_1,2,4,3,5,1\}$	$+\infty$ $()$	1 $(v_1,2)$	3 $(v_1,2,3)$	3 $(v_1,2,4)$	4 $(v_1,2,3,5)$

Table 1.5. *In bold face is indicated the choice made at line 8*

PROOF.– Since the variables X and $T(y)$ are evolving during the execution of the algorithm, we let X_n (respectively, $T_n(y)$) denote the set X (respectively, the value $T(y)$) during the nth iteration. In particular, $X_1 = \{v_1\}$ and $\#X_n = n$ for all $n \leq \#V$. We let v_{n+1} denote the unique vertex in $X_{n+1} \setminus X_n$ selected at line 8. From lines 11–12, observe that either there is no update or there is an update of $T(y)$ and it is replaced by a smaller value. Thus, for all n and all vertices y,

$$T_{n+1}(y) \leq T_n(y). \qquad [1.2]$$

We will show by induction on n that

i) for all $v \in X_n \setminus \{v_1\}$, $T_n(v)$ is the smallest weight of *all* the paths joining v_1 to v;

ii) for all $v \in V \setminus X_n$, $T_n(v)$ is the smallest weight of the paths joining v_1 to v *and* visiting only vertices in X_n before reaching v.

Hence, we will get the expected result when $n = \#V$. We do not take into account the variables $C(y)$ that are simply used to store a path achieving the smallest weight given by $T(y)$.

When $n = 1$, then (i) and (ii) hold from lines 1–5. Assume that (i) and (ii) hold for $1 \leq n < \#V$. We will show that (i) and (ii) hold for $X_{n+1} = X_n \cup \{v_{n+1}\}$.

Proceed by contradiction and assume that (i) does not hold for X_{n+1}. Since (i) holds for X_n this means that (i) does not hold exactly for v_{n+1}: there exists a path $\mathfrak{p}$ joining v_1 to v_{n+1} of weight w less that $T_{n+1}(v_{n+1})$.

But we also know from (ii) that $T_n(v_{n+1})$ is the smallest weight among the paths joining v_1 to v_{n+1} and visiting only vertices in X_n before reaching v_{n+1}.

From [1.2], we have $w < T_{n+1}(v_{n+1}) \leq T_n(v_{n+1})$, thus we deduce that $\mathfrak{p}$ is visiting a vertex $u \notin X_n \cup \{v_{n+1}\}$. Let u be the first vertex of $\mathfrak{p}$ encountered outside of X_n. Note that the first section $\mathfrak{p}'$ of $\mathfrak{p}$ from v_1 to u has weight at most w (the weight of the full path $\mathfrak{p}$).

Using (ii), we get $T_n(u) \leq w$ because $T_n(u)$ is the minimal weight of all paths joining v_1 to u and visiting only vertices in X_n except for the last one and $\mathfrak{p}'$ is a path of this form.

We obtain that $T_n(u) \leq w < T_{n+1}(v_{n+1}) \leq T_n(v_{n+1})$ and thus

$$T_n(u) < T_n(v_{n+1})$$

contradicting the choice of v_{n+1} in line 8 of the algorithm.

We still have to prove (ii) for the $(n+1)$st iteration. How is $T_n(y)$ updated for $y \notin X_{n+1}$ when moving from X_n to X_{n+1}? Consider all paths joining v_1 to y and visiting only vertices in X_{n+1} before reaching y. There are those not going through v_{n+1} and those going through v_{n+1}. For the latter ones, we need only to consider those ending in v_{n+1} because of (i). The conclusion follows from lines 10–12 of the algorithm. ■

1.6. A few applications

In the previous sections, we already have mentioned several applications, e.g. Google's PageRank, graphs associated with social networks like Facebook or Twitter, collaboration graphs and computing shortest path for a GPS device. Of course, graphs occur in many other practical situations: transportation or

flow networks[16], e.g. electrical distribution systems, water running in a series of pipes of different diameters with various capacities and computer networks and distributed resources. We can also think about quivers[17] occurring in the study of friezes in algebraic combinatorics [BER 16, Chapter 10].

Let us present six more examples.

EXAMPLE 1.47 (Group Theory).– *Let $\mathbb{G}$ be a finitely generated group and $g_1, \ldots, g_k$ be generators of $\mathbb{G}$, i.e. every element of $\mathbb{G}$ is a finite product of the g_i's and their inverses. The corresponding* ***Cayley graph*** *of $\mathbb{G}$ is defined as follows: the set of vertices is $\mathbb{G}$ and, for each g_i, there is a directed edge from x to y if and only if $x.g_i = y$. Hence, the outdegree of every vertex is k. In particular, we have an example of an infinite graph whenever $\mathbb{G}$ is infinite. In Figure 1.22, we have represented the Cayley graph[18] of the group of permutations over four elements. This group has 24 elements. We have considered the two cycles $(1\ 2\ 4)$ and $(3\ 4)$ as generators.*

For instance, the label of any closed walk is a word over the generators and their inverses, which is equal to the identity element of the group; taking an edge backward corresponds to multiplication by the inverse of a generator. More generally, the labels of two walks between any two vertices are two representations of the same element of the group. The multiplication by the cycle $(1\ 2\ 4)$ (respectively,$(3\ 4)$) is represented by a gray (respectively, black) edge. The identity permutation is the vertex 1.

EXAMPLE 1.48 (Combinatorics on words).– *In combinatorics on words, we are interested in properties of infinite words, i.e. maps $\mathbf{w}$ from $\mathbb{N}$ to a finite set A called alphabet. In particular, we can search for the set of factors made up of n consecutive symbols*

$$\mathsf{Fact}_{\mathbf{w}}(n) := \{\mathbf{w}(i)\cdots\mathbf{w}(i+n-1) \mid i \geq 0\}$$

that may occur in $\mathbf{w}$. The celebrated Thue–Morse word [ALL 99] starts with

$$01101001100101101001011001101001\cdots$$

16 In this book, we will not discuss this important topic; for a few pointers make a search about Ford–Fulkerson algorithm or max-flow min-cut theorem.

17 This terminology simply refers to a directed multigraph and can be encountered in category theory and representation theory.

18 Such a graph can be easily obtained using `Mathematica` with a built-in function `CayleyGraph` or other softwares like `SAGE`.

and does not contain any cube, i.e. a factor of the form uuu*. Thus, the factors of length* 3 *occurring in this word are* 001, 010, 011, 100, 101, 110*. A* Rauzy graph *is a handy tool to study the structure of these factors. The set of vertices of the Rauzy graph of order* $n > 1$ *is* $\mathsf{Fact}_{\mathbf{w}}(n)$ *and there is an edge labeled by* b *from* u *to* v *if* $u = ax$ *and* $v = xb$ *where* a, b *are symbols and* x *is a word of length* $n - 1$ *such that* axb *is a factor occurring in* $\mathbf{w}$*. Thus, we learn from the Rauzy graph some information about the sequencing of the factors of length* n *within* $\mathbf{w}$*. More about these graphs are presented in section 3.6. We will see that a Rauzy graph is a subgraph of a de Bruijn graph. In Figure 1.23, we have depicted the Rauzy graph of order* 3 *of the Thue–Morse word.*

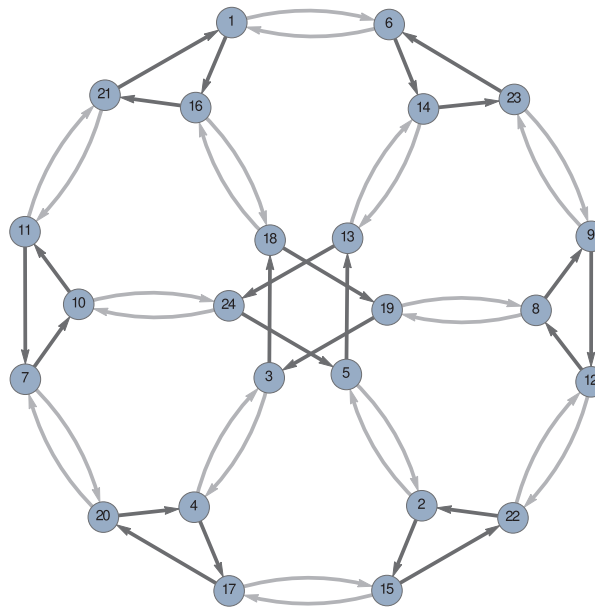

Figure 1.22. *A Cayley graph for* $\mathcal{S}_4$

EXAMPLE 1.49 (Chemistry).– *Several molecules may have the same formula but distinct molecular structures. They are called* isomers*: they have the same number of atoms but with different arrangements so they can have different properties. We can ask which configurations are possible from a combinatorial point of view (see [PÓL 87, TEM 96]).*

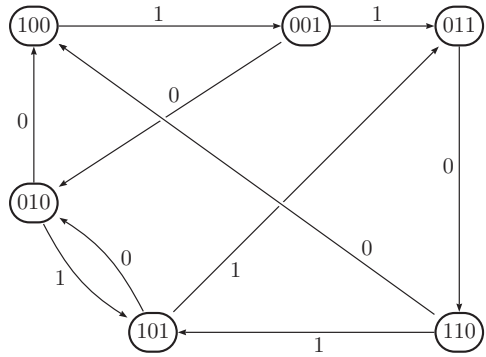

Figure 1.23. *The Rauzy graph of order* 3 *for the Thue–Morse word*

EXAMPLE 1.50 (Assigning IP Addresses).– *The assignment of IP addresses to nodes in a network takes into account the natural hierarchy of the network and its division into subnetworks. Assignment of addresses then follows a simple rule inside these subnetworks. The goal is to minimize the sizes of routing tables on the nodes and packets are forwarded using a longest prefix matching.*

For a related (but quite unrealistic) example, we may assign addresses to the vertices of a simple graph G in such a way that the Hamming distance[19] *between two addresses is equal to the distance between the two vertices if and only if [DJO 73]:*

– G is a connected bipartite graph;

– for every edge $\{a, b\}$ of G, if for all vertices x, y, z such that $\mathsf{d}(a, x) < \mathsf{d}(b, x)$, $\mathsf{d}(a, y) < \mathsf{d}(b, y)$, $\mathsf{d}(x, z) + \mathsf{d}(z, y) = \mathsf{d}(x, y)$, then $\mathsf{d}(a, z) < \mathsf{d}(b, z)$.

EXAMPLE 1.51 (Coloring).– *Here is an application of the notion of proper coloring introduced in example 2.10. Consider a train transporting several chemical products. In case of a train accident, it is important that some of these products do not mix because it would lead to producing toxic or explosive reactions. The chemist tells us which products may or may not mix. It is the task of the organizer to put products that may not mix in different wagons. But from an economical point of view, we also want to minimize the*

19 The Hamming distance between two finite sequences $u, v \in \{0, 1\}^k$ is the total number of the indices i such that $u_i \neq v_i$.

number of wagons. Assume that we are transporting products $P_1, P_2, \ldots, P_9$. The products are the vertices of the graph depicted in Figure 1.24. There is an edge between two products if they may not be mixed. Chapter 7 presents some results on colorings. Here, three wagons are needed because of the cycle of length 3. These wagons are enough: a first wagon for P_5, P_7, P_8, a second wagon for P_1, P_2, P_3, P_6 and a third wagon for P_4 and P_9. Every wagon is made up of a set of pairwise independent vertices. This is not the only solution. Of course, you can also think of extra constraints depending on the quantities to be carried.

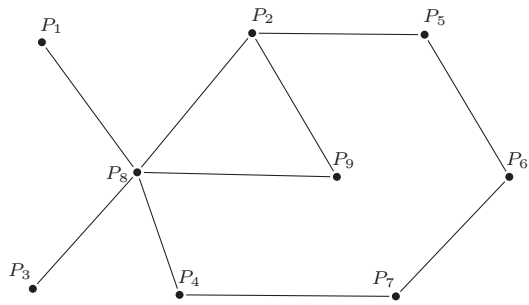

Figure 1.24. *Chemical products that may not be mixed*

EXAMPLE 1.52 (Detecting communities in large graphs).– *The question of community detection is to find a partition of a network into "communities" of densely connected nodes, with the nodes belonging to different communities being only sparsely connected. I agree that this is a non-rigorous definition. There are several algorithms that give partitions of this form. The quality of the resulting communities can, for instance, be measured by its* modularity, *a real number in $[-1, 1]$ defined by*

$$\frac{1}{2M} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2M} \right] \delta(c_i, c_j)$$

where A_{ij} is the weight between the vertices i and j (weights can express stronger links between some vertices, for instance the number of coauthored publications in a collaboration graph), $k_i = \sum_j A_{ij}$, c_i is the community to which vertex i is assigned by the algorithm and $M = \frac{1}{2} \sum_{ij} A_{ij}$. As usual $\delta(x, y) = 1$ if $x = y$ and 0 otherwise. To give a few pointers, see [BLO 08] and also [NEW 06, NEW 04, PON 11, PON 06].

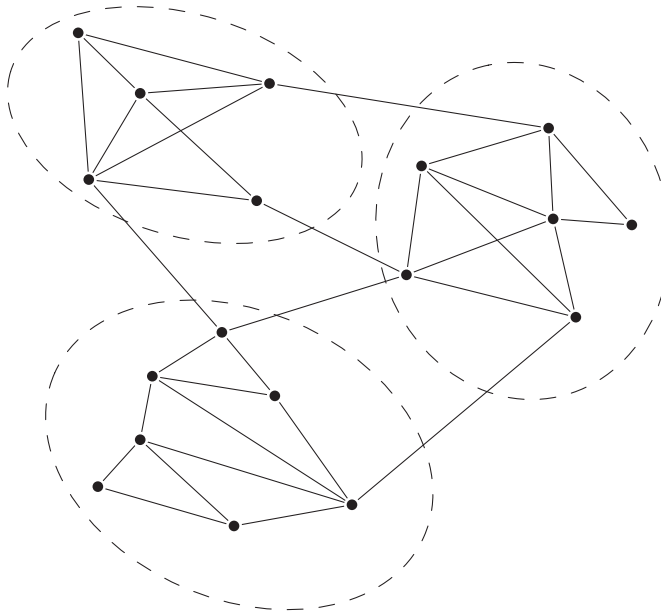

Figure 1.25. *Three communities in a graph*

1.7. Comments

We give some comments or remarks in chronological order of appearance within the text.

Generalizations of multisets have been proposed, for instance, fuzzy multisets [YAG 87] or real-valued multisets [BLI 89].

A generalization of a graph is given by a *hypergraph* $H = (V, E)$ where V is, as usual, the set of vertices and E is the set of *hyperedges*, i.e. a subset of the set of non-empty subsets of V. We can therefore model other types of relations occurring between more than two vertices. Several variants exist (see [BER 89]).

For some general references on graph theory, see [DIE 10] or [BON 08]. In particular, the first reference gives more details on infinite graphs.

About ranking and rating techniques discussed in example 1.6, we will study the basics of the PageRank algorithm in Chapter 10. For a pleasant and comprehensive introduction to the subject, see [LAN 12].

The *road coloring problem* mentioned in example 1.15 can be stated as follows: given a k-regular irreducible aperiodic[20] digraph, is it possible to color the edges in such a way that there exists a synchronizing sequence $\mathfrak{s}$, i.e. there exists a vertex v such that, for all vertices u, following a path of label $\mathfrak{s}$ from u leads to v. This problem was first considered by Adler, Goodwyn and Weiss [ADL 77] and solved by Trahtman [TRA 09]. This problem has applications in data storage [LIN 95], automated design [EPP 90] or communication protocols [AHO 95] (see the chapter by Béal and Perrin in [BER 16]).

In a *small world*, everyone on earth is supposedly connected to everyone else by a chain of at most six "friends". The famous psychologist Milgram and his collaborators started to examine this phenomenon with people asked to send letters across the United States, see [EAS 10, p. 31]. Graph theoretic models may also explain other social phenomena such as homophily and the *glass ceiling effect* that keeps women from reaching highest positions in companies [AVI 15]. *Homophily* is the tendency for people to stay "close" to other people sharing the same characteristics (e.g. gender, ethnicity and cultural tastes), see again [EAS 10] and Schelling's mathematical model of segregation in sociology [SCH 71, ZHA 04].

Concerning *Eulerian graphs*, when a multigraph is not Eulerian we can try to find a closed walk of minimal length that visits every edge at least once. This problem is known as the *Chinese postman problem*. The number of Eulerian circuits in a connected Eulerian digraph (the situation is more difficult in the unoriented case) is given by the so-called BEST theorem named after Ehrenfest and de Bruijn [VAN 51], Tutte and Smith [TUT 41]

$$t_w(G) \prod_{v \in V(G)} \big(\mathsf{deg}^+(v) - 1\big)!$$

where we recall that for an Eulerian digraph $\mathsf{deg}^+(v) = \mathsf{deg}^-(v)$ for all v (see corollary 1.43) and $t_w(G)$ denotes the number of arborescences rooted at the vertex w. An *arborescence* rooted at w is a digraph where, for every vertex v, there is exactly one path from w to v. We will see in section 8.6 (and more

20 Aperiodicity will be discussed in Chapter 9.

precisely, with the concluding corollary 8.44) that $t_w(G)$ does not depend on the choice of w. As an example, consider the graph depicted in Figure 1.26, we can check that it contains three Eulerian circuits. We have also depicted the three arborescences rooted at the bottom left vertex. Applying the above formula yields

$$t_w(G).1!.1!.1!.0! = 3.$$

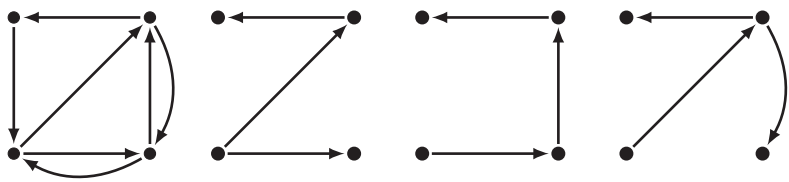

Figure 1.26. *Number of Eulerian circuits and arborescences*

Domination (definition 1.23) in graph is an important topic of its own. See, for instance, [HEN 13] for more on the subject. For general graphs, the determination of the total domination number is an NP-complete problem [PFA 83] (see also example 2.12).

For important practical issues and implementation of Dijkstra's algorithm, several refinements have been considered. For details and discussions about the average-case analysis of this algorithm, see, for instance, [MEH 08].

Connected to Cayley graphs, the *word problem for groups* is a well-known question arising in abstract algebra: given any two words written over an alphabet of generators and their inverses, can we algorithmically decide whether these two words represent the same element of the group? In full generality, Novikov showed that this problem is undecidable [NOV 55]. For special families of groups, e.g. automatic groups, the problem is decidable.

1.8. Exercises

1) Can we find a group of 11 people such that each member of the group exactly knows three other people belonging to the group? Same question but with a group of eight people. In case of a positive answer, draw a graph illustrating the situation. Is such a graph always connected?

2) In a meeting with at least two persons, people who know each other shake hands (and no one else). Prove that there are two persons who shaked exactly the same number of hands.

3) n couples are invited to a party. Some guests shake hands, but nobody shake hands with his/her partner. One of the guests, Mr. G., asks all guests how many hands they have shaken. He gets $2n-1$ different answers. How many hands has the wife of Mr. G. shaken?

4) Prove that there is no simple graph that is 3-regular and has seven vertices. Prove that there is no graph with an odd number of vertices and all vertices of odd degree.

5) Let G be a simple graph whose vertices have degree at least 2. Prove that G has a cycle.

6) Let $G=(V,E)$ be a connected graph. We make use of the notion of distance (remark 1.31). The *eccentricity* of a vertex u is defined by

$$\epsilon(u) := \max_{v\in V} \mathsf{d}(u,v).$$

As usual in a metric space, the *diameter* of G (also see definition 8.32) is

$$\mathsf{diam}(G) := \max_{u\in V} \epsilon(u).$$

The *radius* of G is defined as

$$\mathsf{rad}(G) = \min_{u\in V} \epsilon(u) = \min_{u\in V}\max_{v\in V} \mathsf{d}(u,v).$$

Prove that

$$\mathsf{rad}(G) \leq \mathsf{diam}(G) \leq 2\,\mathsf{rad}(G).$$

For which graphs do we have $\mathsf{rad}(G) = \mathsf{diam}(G)$? Same question with $\mathsf{diam}(G) = 2\,\mathsf{rad}(G)$.

7) Let $G=(V,E)$ be a connected graph. Let $k = \max_{v\in V} \mathsf{deg}(v)$. If $k \geq 3$, prove that

$$\#V \leq \frac{k(k-1)^{\mathsf{rad}(G)}}{k-2}.$$

8) Prove that a minimum vertex cover of the Petersen has size 6, see Figure 1.8.

9) Give an example of a Hamiltonian graph that is not Eulerian.

10) Prove that the Petersen graph (depicted in Figure 1.8) is not Hamiltonian, but find a Hamiltonian path.

11) Prove that the complete bipartite graph $K_{m,n}$ is Hamiltonian if and only if $m = n$.

12) Let $n \geq 1$. The *n-cube* Q_n is the graph defined below. First the graphs Q_1, Q_2 and Q_3 have been represented in Figure 1.27.

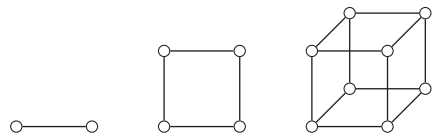

Figure 1.27. *The graphs Q_1, Q_2 and Q_3*

For all $n \geq 1$, we get Q_{n+1} by considering two disjoint copies of Q_n and adding an edge for every pair of vertices that correspond to each other in the two copies of Q_n. A representation of Q_4 is given in Figure 1.28.

a) In terms of n, how many vertices and edges do Q_n have?

What is the degree of every vertex in Q_n?

b) For which values of n, is the n-cube Hamiltonian?

c) For which values of n, is the n-cube Eulerian?

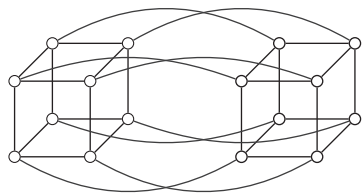

Figure 1.28. *A representation of Q_4*

13) Let $n \geq 1$. Define the graph where the vertices are 2^n strings of length n over $\{0, 1\}$ and two vertices are connected if and only if their Hamming distance (see example 1.50) is one. Compare this graph with the n-cube introduced above.

14) For the n-cube, determine its edge-connectivity $\lambda(Q_n)$.

15) Prove that a (multi)graph G is bipartite if and only if every circuit in G has an even length.

16) Find all values a, b, c with $1 \leq a \leq b \leq c$ such that the complete tripartite graph $K_{a,b,c}$ has a Eulerian trail but no Eulerian circuit.

17) Give examples of simple graphs such that $\lambda(G) = i$ for $i = 1, 2, 3, 4$. Same question with vertex-connectivity $\kappa(G)$.

18) Use Menger's theorem to derive a polynomial time algorithm computing the vertex-connectivity of a graph.

19) Prove that if $F \subset E(G)$ is a minimal cut-set of a connected graph G, i.e. for all $f \in F$, $F \setminus \{f\}$ is not a cut-set, then the number of connected components of $G - F$ is exactly 2.

20) Prove that a 3-regular graph has a cut-vertex if and only if it has a bridge.

21) Let G be a simple graph with m edges $e_1, \ldots, e_m$. We define the **line graph** $L(G)$ as the graph with m vertices $v_1, \ldots, v_m$ and the edge $\{v_i, v_j\}$ belongs to $L(G)$ if and only if the edges e_i and e_j of G are adjacent (i.e. they share an endpoint).

a) represent the line graph of the complete graph K_4, the bipartite complete graph $K_{2,3}$ and a cycle with six vertices,

b) show that $K_{1,3}$ and K_3 have the same line graph,

c) express the number of edges in $L(G)$ in terms of the degrees of the vertices of G,

d) show that if G is a simple k-regular graph (i.e. each vertex has degree k), then $L(G)$ is $(2k-2)$-regular.

22) Build a simple 3-regular graph that has a cut-edge. Determine the minimal number of vertices that such a graph has.

23) Work out a proof of the exactness of Fleury's algorithm.

24) Prove the following [BON 69]. Let G be a graph with n vertices ordered by increasing degree $\deg(u_1) \leq \deg(u_2) \leq \cdots \leq \deg(u_n)$. This sequence is called the *degree sequence*. If there exists $k \in \{0, \ldots, n\}$ such that $\deg(u_j) \geq j + k - 1$ for $j = 1, \ldots, n - 1 - \deg(u_{n-k+1})$, then G is k-connected (i.e. k-vertex connected).

25) With the notation of definition 1.38, prove the following [CHA 68]. Let $G \neq K_n$ be a graph with n vertices. Then, $\kappa(G) \geq 2 \min_{v \in V(G)} \deg(v) + 2 - n$.

26) Consider the Roy–Warshall algorithm described in Table 1.6. The input is a simple digraph with n vertices $\{1, \ldots, n\}$ given by its adjacency matrix $\mathbf{A}(G)$. In the last line of the algorithm, the evaluation $\mathbf{M}_{i,j}$ or $(\mathbf{M}_{i,k}$ and $\mathbf{M}_{k,j})$ returns 1 if $\mathbf{M}_{i,j} = 1$ or if both $\mathbf{M}_{i,k} = 1$ and $\mathbf{M}_{k,j} = 1$. Recall that we write $u \to v$ if u is connected to v. Prove that this cubic-time algorithm (with respect to n) returns a matrix $\mathbf{M}$ where $\mathbf{M}_{u,v} = 1$ if and only if $u \to v$. Hence, this matrix provides the connectivity relation within G. An application of the algorithm is depicted in Figures 1.29 and 1.30.

ROY–WARSHALL($\mathbf{A}(G)$)

1 $\mathbf{M} \leftarrow \mathbf{A}(G)$;
2 **for** $i = 1$ **to** n
3 **do** $\mathbf{M}_{i,i} \leftarrow 1$
4 **for** $k = 1$ **to** n
5 **do for** $i = 1$ **to** n
6 **do for** $j = 1$ **to** n
7 **do** $\mathbf{M}_{i,j} \leftarrow \mathbf{M}_{i,j}$ or $(\mathbf{M}_{i,k}$ and $\mathbf{M}_{k,j})$

Table 1.6. *Roy-Warshall algorithm for connectivity*

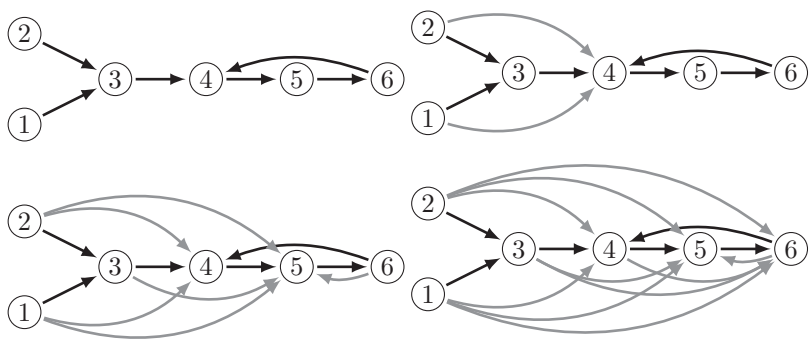

Figure 1.29. *Application of Roy–Warshall algorithm for* $k = 3, 4, 5$

27) What can be said about an infinite word $\mathbf{w}$ whose Rauzy graph (see example 1.48) of order n is reduced to a cycle? What can be said about Rauzy graphs of order n for ultimately periodic infinite words, i.e. words of the form $uvvv\cdots$ where u, v are finite non-empty words. *Sturmian words* are

infinite words $\mathbf{w}$ characterized by $\#\mathsf{Fact}_{\mathbf{w}}(n) = n + 1$ for all $n \geq 0$. Can you characterize Sturmian words using Rauzy graphs?

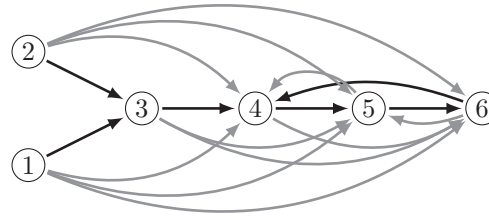

Figure 1.30. *Final application of Roy–Warshall algorithm for* $k = 6$