# Chapter 1
# Basic concepts

## 1.1 Graphs

In this opening chapter it is our intention to give the basic definitions and some of the notation which we shall use throughout this book. We begin with some basic concepts concerning graphs.

A *graph* is an ordered triple $(V(G), E(G), \psi_G)$ consisting of a finite, non-empty set of *vertices*, $V(G)$, a finite set $E(G)$ of *edges*, disjoint from $V(G)$, and an *incidence function* $\psi_G$ that associates an unordered pair of distinct vertices with each edge. We say that $e$ joins $u$ and $v$ if $\psi_G(e) = \{u, v\}$, written $uv$, and that $e$ has *ends* $u$ and $v$. An edge is *incident* with a vertex $v$ if $v$ is one of its ends, and two vertices joined by an edge are *adjacent*. Edges which have the same ends are called *multiple* or *parallel edges*: the case when an edge joins a vertex to itself will not be considered in this book. We shall call a graph with no multiple edges *simple*. If $G$ is simple and $e \in E(G)$ an edge with $\psi_G(e) = uv$, we shall write $e = uv$. The *degree* $d_G(v)$ of a vertex $v$ is the number of edges incident with $v$. We denote the minimum and maximum degrees of $G$ by $\delta(G)$ and $\Delta(G)$, respectively. A vertex with degree zero is called an *isolated vertex*. A graph with only one vertex is called *trivial*, and all other graphs *non-trivial*. A graph is *regular* if every vertex has the same degree, and it is *$k$-regular* if that degree is $k$. A 3-regular graph is called a *cubic* graph. A graph $G$ is *planar* if it can be drawn in the plane so that its edges intersect only at their ends.

A graph $H$ is a *subgraph* of another graph $G$ if $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$ and $\psi_H$ is the restriction of $\psi_G$ to $E(H)$. We denote this by $H \subseteq G$. When $H \subseteq G$ without equality we say that $H$ is a *proper subgraph* of $G$ and write $H \subset G$. $H$ is a *spanning subgraph* of $G$ when $V(G) = V(H)$. Suppose now that $V \subseteq V(G)$ is a subset of vertices. We may consider the subgraph of $G$ with vertex set $V$, and with edge set consisting of all those edges which have

both ends in $V$. We call this the subgraph *induced* by $V$ and write it as $G[V]$. Similarly we can define a subgraph induced by a subset of edges $E \subseteq E(G)$; this graph will have edge set $E$ and a vertex set consisting of all the end vertices of edges in $E$. This we denote by $G[E]$. Two subgraphs $H_1$ and $H_2$ of $G$ are said to be *vertex-disjoint* or simply *disjoint* if $V(H_1) \cap V(H_2) = \emptyset$, and similarly $H_1$ and $H_2$ are *edge-disjoint* subgraphs if $E(H_1) \cap E(H_2) = \emptyset$.

A *walk* in a graph $G$ is a finite sequence of vertices and edges, $W = v_0e_1v_1 \dots e_nv_n$ where $\psi_G(e_i) = v_{i-1}v_i$ for each $1 \leq i \leq n$. When it is clear which edges are involved we shall write the walk $W = v_0e_1v_1 \dots e_nv_n$ as simply $W = v_0v_1 \dots v_n$. A walk from $u$ to $v$ we shall call a $(u, v)$-*walk* and by the *length* of a walk we mean the number of edges which are traversed. We say that a non-trivial $(u, u)$-walk is *closed*. Given a $(u, v)$-walk $W$, the $(v, u)$-walk obtained by traversing $W$ in the opposite direction is denoted by $W^{-1}$. A *trail* is a walk in which the edges are distinct and a *path* is a walk in which the edges and vertices are distinct. A closed trail, in which the origin and internal vertices are distinct, is called a *cycle*. We shall also use the words 'path' and 'cycle' to denote the graph or subgraph whose vertices and edges are those of a path or a cycle, respectively. The cycle of length 3 is called a *triangle*. A graph without triangles is called *triangle-free*.

Two vertices $u$ and $v$ of $G$ are *connected* if there is a $(u, v)$-path in $G$. 'Connection' is an equivalence relation on the set of vertices. Thus, we can form a partition of the vertex set $V(G) = V_1 \cup \dots \cup V_n$ so that two vertices are connected if and only if they are from the same subset. Then the subgraphs $G[V_i]$ are called the *connected components* of $G$, and $G$ is called *connected* if it has only one such component.

In a connected graph, we may consider the length of the shortest path between two vertices $u$ and $v$. This length is called the *distance* between the $u$ and $v$, $d_G(u, v)$ or simply $d(u, v)$. We write $N_{i,G}(v)$, or simply $N_i(v)$, for the set of vertices at distance $i$ from $v$. $N_{1,G}(v)$ is called the *neighbourhood* of $v$; we shall write $N_G(v)$, $N_1(v)$, or even $N(v)$, for the neighbourhood of $v$, when the context is clear. Analogously for a subset of vertices $S \subset V(G)$ we denote the set of all vertices of $G$ which are adjacent to at least one vertex in $S$ by $N_G(S)$, or simply $N(S)$. Let $k = \max\{i : N_i(v) \neq \emptyset\}$, then we call the partition of the vertex set $V(G) = N_0(v) \cup N_1(v) \cup \dots \cup N_k(v)$ the *level representation of $G$ with respect to $v$*. Algorithmically this partition is easily constructed, by inductively defining $N_i(v)$ to be the set of neighbours of $N_{i-1}(v)$ which have not already been accounted for. If $X$ and $Y$ are disjoint subsets of vertices in a graph $G$ then we write $E_G(X, Y)$, or simply $E(X, Y)$, for the set of edges with one end in $X$ and the other in $Y$. If $X = \{x\}$ then we write $E(x, Y)$ for $E(\{x\}, Y)$.

We also need some operations to combine and change graphs. Given two graphs $G$ and $H$ we denote by $G \cup H$ the graph with vertex set $V(G) \cup V(H)$, and edge set $E(G) \cup E(H)$. Correspondingly, if $X \subset V(G)$ and $Y \subset E(G)$,

we write $G - Y$ for the graph obtained by removing $Y$ from the edge set of $G$ and $G - X$ for $G[V(G)\backslash X]$. In the cases when $X = \{v\}$ and $Y = \{e\}$ we write $G - v$ and $G - e$, respectively, for $G - \{v\}$ and $G - \{e\}$. We denote the graph obtained from $G$ by joining two non-adjacent vertices $u$ and $v$ by $G + uv$. An edge $e$ is said to be ***subdivided*** when it is deleted and replaced by a path of length 2 connecting its ends, the internal vertex of this path being a new vertex. The bipartite graph obtained from a graph $G$ by subdividing every edge is denoted by $bip(G)$. The (***lexicographic***) ***product*** of the disjoint simple graphs $G$ and $H$ is the graph denoted by $G \times H$ with vertex set $V(G \times H)$ consisting of all ordered pairs $(u, v)$ with $u \in V(G)$ and $v \in V(H)$, and with edge set $E(G \times H) = \{(u_1, v)(u_2, v) : u_1u_2 \in E(G), v \in V(H)\} \cup \{(u, v_1)(u, v_2) : v_1v_2 \in E(H), u \in V(G)\}$. Although there are many others (see Sabidussi (1960)), this is the only graph product we shall consider.

We say that two graphs $G$ and $H$ are ***isomorphic*** if there are two bijections $\theta : V(G) \longleftrightarrow V(H)$ and $\phi : E(G) \longleftrightarrow E(H)$ so that $\psi_G(e) = uv$ if and only if $\psi_H(\phi(e)) = \theta(u)\theta(v)$. In particular, when $G$ and $H$ are simple graphs, they are isomorphic if there is a bijection $\theta : V(G) \longleftrightarrow V(H)$ so that $uv \in E(G)$ if and only if $\theta(u)\theta(v) \in E(H)$.

A ***directed graph*** $D$ is also an ordered triple $(V(D), E(D), \xi_D)$ where $V(D)$ is a non-empty set of vertices, $E(D)$ is a set of ***arcs*** or ***directed edges*** (disjoint from $V(D)$) and $\xi_D$ is an incidence function which associates an ordered pair of distinct vertices with each edge in $E(D)$. Let $v_0e_1v_1e_2v_2 \dots e_nv_n$ be a sequence where $v_0, \dots, v_n$ are vertices, $e_1, \dots, e_n$ are arcs, and $\xi_D(e_i) = (v_{i-1}, v_i)$ for each $i = 1, \dots, n$. If $v_0, v_1, \dots, v_n$ are distinct this sequence is called a ***directed path***; if $v_0 = v_n$ and $v_1, \dots, v_n$ are distinct the sequence is a ***directed cycle***. From each graph $G$ we can obtain a directed graph by specifying, for each edge, an order of its ends. We call such a directed graph an ***orientation of*** $G$ and denote it by $\vec{G}$. On the other hand from each directed graph $D$ we can obtain a graph by ignoring the directions of the edges; this graph is called the ***underlying graph*** of $D$. If the underlying graph of $D$ is simple and $e$ is an arc of $D$ with $\xi_D(e) = (u, v)$ then we denote $e$ by $\overrightarrow{uv}$. In a natural way all the concepts we have defined for graphs have their counterparts for directed graphs by considering their underlying graphs. If $v$ is a vertex of $D$ then we denote the set of neighbours of $v$ adjacent to $v$ via an arc starting at $v$ by $N^+(v)$ and that via one ending at $v$ by $N^-(v)$. The cardinalities of $N^+(v)$ and $N^-(v)$ are denoted by $d^+_G(v)$ and $d^-_G(v)$ respectively.

The terminology above is broadly consistent with that used in a variety of other books. In particular, that used in the book of Bondy and Murty (1976) is perhaps the closest.

## 1.2 Partially ordered sets

A finite *partially ordered set* (or *poset*) $\mathcal{P} = (P, \succeq)$ consists of a finite set $P$ and a relation $\succeq$ which satisfies

$$(1) \quad x \succeq x, \forall x \in P \qquad \text{(reflexivity)},$$
$$(2) \quad x \succeq y \text{ and } y \succeq x \Rightarrow x = y, \forall x, y \in P \qquad \text{(antisymmetry)},$$
$$(3) \quad x \succeq y \text{ and } y \succeq z \Rightarrow x \succeq z, \forall x, y, z \in P \qquad \text{(transitivity)}.$$

If $x \succeq y$ and $x \neq y$ then we write $x \succ y$. If $x \succ y$ and there is no element $w \in P$ with $x \succ w \succ y$ then we say that $x$ *covers* $y$. An element $x$ is called *maximal* in $P$ if there exists no element $y$ with $y \succ x$. Using the cover relation we can define a graphical representation of $\mathcal{P}$. The *Hasse diagram* of $\mathcal{P}$ is the directed graph $\vec{G}$ with vertex set $P$ where vertices $x$ and $y$ are joined by an directed edge $\overrightarrow{xy}$ if $x$ covers $y$ in $\mathcal{P}$. The underlying simple graph $G$ is called the *non-oriented Hasse diagram* of $\mathcal{P}$.

The *greatest lower bound* of a subset of elements $T \subseteq P$ is an element $g$ such that $x \succeq g$ for every $x \in T$ and for any $h \in P$ with $x \succeq h$ for every $x \in T$, we have $g \succeq h$. If such a greatest lower bound exists, its uniqueness is guaranteed by the antisymmetry of $\succeq$. Similarly the *least upper bound* of a subset $T \subset P$, if it exists, is an element $l$ such that $l \succeq x$ for every $x \in P$, and for any $k$ with $k \succeq x$ for every $x \in P$ we have $k \succeq l$.

A finite *lattice* is a finite partially ordered set, in which every two elements have a greatest lower bound, and a least upper bound. A finite *modular lattice* is a lattice satisfying the additional conditions that for every $x, y \in P$, the least upper bound of $\{x, y\}$ covers $x$ and $y$, and both $x$ and $y$ cover the greatest lower bound of $\{x, y\}$. Finally, the *boolean lattice* with $2^n$ elements is the partially ordered set consisting of all subsets of some $n$-element set ordered by inclusion.

## 1.3 Reducibility of problems and NP-completeness

The questions in discrete mathematics which we might wish answered are many and various, but most can be rephrased into questions which require a 'YES' or 'NO' answer. Such questions are called *decision problems*, and they normally concern some object or *input*. For instance, we might ask

Problem 1: *Can the vertices of a graph be each labelled with one of two colours, so that no edge has both its ends labelled with the same colour?*
or

Problem 2: *Does the graph $G$ have a cycle which includes every vertex?*

In each of these questions the graph $G$ is the input to the problem. There are various ways of representing this input graph to perhaps some computer, for instance as a collection of vertices and edges or as an adjacency matrix, but we shall envisage this representation as being coded as a binary string in some fashion. This done, we can identify the question with a collection of binary strings – the collection of binary strings which represent graphs for which the answer to the question is 'YES'. More formally, given a decision problem $\mathcal{D}$ we define the *language* $\mathcal{L}(\mathcal{D})$ to be the set of input strings to $\mathcal{D}$ for which the answer to $\mathcal{D}$ is 'YES'. In other words the decision problem $\mathcal{D}$ could be thought of as the problem of deciding whether a given input string is a member of the language $\mathcal{L}(\mathcal{D})$ or not.

The two problems we posed above are of somewhat different natures. As we shall see later in this volume, there is a rather efficient algorithm to answer the first, but no such algorithm is known for the second. We can formalise the distinction in terms of the recognition of binary strings.

We say that a decision problem belongs to the *P class* (is a P problem) if there are an algorithm $\mathcal{A}$ and a number $\alpha$ so that for every binary input string of length $\beta$, the algorithm $\mathcal{A}$ will have decided whether the input string is a member of $\mathcal{L}(\mathcal{D})$ after $\beta^{\alpha}$ operations. Such an algorithm, which always stops after a number of operations which is a polynomial function of the length of the input, is said to operate in *polynomial time* and is called a *polynomial algorithm*.

We say (see also Wilf (1986)) that a decision problem $\mathcal{D}$ belongs to the *NP class* (is an NP problem) if there is a polynomial algorithm $\mathcal{A}$ which carries out the following:

*(1)* For every input string $I$ which is in the language $\mathcal{L}(\mathcal{D})$ there is a *certificate* string $C(I)$ so that when $I$ and $C(I)$ are input to $\mathcal{A}$, the algorithm recognises that $I$ is in $\mathcal{L}(\mathcal{D})$.

*(2)* If some input string $I$ does not belong to the language $\mathcal{L}(\mathcal{D})$ then there is no certificate string $C(I)$ which will cause $\mathcal{A}$ to conclude that $I$ is a member of $\mathcal{L}(\mathcal{D})$.

We shall see later that Problem 1 is a P problem, but that both Problems 1 and 2 can be easily shown to be members of the NP class. For Problem 1 the certificate might be a labelling of the vertices with colours: the algorithm $\mathcal{A}$ then only has to check that each vertex of the graph is labelled with one of only two colours and that no edge has both its ends labelled with the same colour – a task which can easily be carried out in a polynomial number of operations. In a similar way a sequence of vertices can be a certificate for Problem 2. For to check that this sequence actually describes a cycle in the graph which passes through each vertex is an easy task. It is perhaps worth pointing out that to find a certificate which verifies the 'YES' answer to some NP problem may be very difficult, but to establish that the problem is

an NP problem requires only that the certificate can be shown to be valid in polynomial time. More generally, it is clear that P⊆NP, but the question as to whether this containment is strict remains as yet unanswered. A problem $\mathcal{D}$ is a member of the *co-NP class* (is a co-NP problem) if the problem of deciding whether the answer to $\mathcal{D}$ is 'NO' is an NP problem.

It is often the case that a solution to one problem also provides solutions to many others, by employing some judicious rephrasing or reformulation. Consequently, it is natural to define some formal sense of reformulation for formal decision problems.

We say, of two decision problems $\mathcal{D}$ and $\mathcal{E}$, that $\mathcal{D}$ is *polynomially reducible* to $\mathcal{E}$, if there is a polynomial algorithm $\mathcal{A}$ which, given an input string to $\mathcal{D}$, constructs an input string for $\mathcal{E}$ which has the same answer ('YES' or 'NO'). We say that a problem is *NP-complete* if it itself is an NP problem, and every other NP problem can be polynomially reduced to it. Thus in some sense NP-complete problems are the hardest NP problems of all. Indeed polynomial reducibility can be defined not only for decision problems, but also for wider classes of problems. We shall say that a problem is *NP-hard* if any NP problem can be polynomially reduced to it. For instance, the problem of deciding whether, in a graph $G$, there is a subset $C$ of at most $k$ vertices such that any edge of $G$ is incident with some vertex in $C$ is NP-complete (Karp (1972)), but the problem of finding the smallest such $C$ is NP-hard.

Clearly to show that a particular problem is NP-complete it is enough to show that some other NP-complete problem is polynomially reducible to it. Several hundreds of NP-complete problems from all over mathematics have been identified (see Garey and Johnson (1979)). As far as bipartite graphs go, we give a list of NP-complete problems in the Appendix at the back of this book.

# Chapter 2
# Introduction to bipartite graphs

## 2.1 Recognising bipartite graphs

A graph $G$ is *bipartite* if the vertex set $V(G)$ can be partitioned into two sets $V_1$ and $V_2$ in such a way that no two vertices from the same set are adjacent. The sets $V_1$ and $V_2$ are called the *colour classes* of $G$ and $(V_1, V_2)$ is a *bipartition* of $G$. In fact a graph being bipartite means that the vertices of $G$ can be coloured with at most two colours, so that no two adjacent vertices have the same colour. Throughout this book we will depict bipartite graphs with their vertices coloured black and white to show one possible bipartition. We shall call a graph $m$ *by* $n$ *bipartite*, if $|V_1| = m$ and $|V_2| = n$, and a graph a *balanced bipartite graph* when $|V_1| = |V_2|$; for example the graph shown in figure 2.1.1 is a 4 by 4 bipartite graph, since figure 2.1.2 shows that it has a bipartition in which each of the colour classes has four vertices. Given these basic definitions, let us begin by making some simple observations about the structure of bipartite graphs.

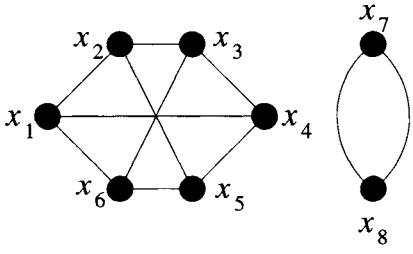


Figure 2.1.1

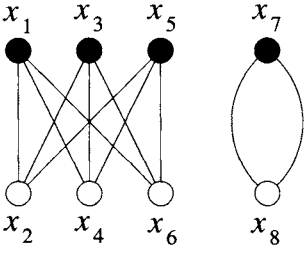


Figure 2.1.2

**Property 2.1.1** *A connected bipartite graph has a unique bipartition.*

**Property 2.1.2** *A bipartite graph, without isolated vertices, which has $t$ connected components has $2^{t-1}$ bipartitions.*

For example, the bipartite graph in figure 2.1.2 has two bipartitions. One is, of course, shown in the figure and the other has $V_1 = \{x_1, x_3, x_5, x_8\}$ and $V_2 = \{x_2, x_4, x_6, x_7\}$.

During the course of this book we shall meet several characterisations of bipartite graphs, but let us begin with one of the most widely used, which was obtained by König (1916).

**Theorem 2.1.3** *A graph $G$ is bipartite if and only if $G$ has no cycle of odd length.*

**Proof.** Suppose that $G$ is a bipartite graph with bipartition $(V_1, V_2)$ and $C = v_0v_1v_2 \dots v_kv_0$ is a cycle of $G$. Without loss of generality, we may assume that $v_0 \in V_1$. Then, since $G$ is bipartite, $v_1$ must be a vertex of $V_2$. Indeed we must have that $v_{2i} \in V_1$ and $v_{2i+1} \in V_2$. Hence $k$ must be odd, and $C$ is an even cycle.

Clearly it suffices to prove the converse when $G$ is connected. Let $G$ contain only even cycles, and let $v$ be an arbitrary vertex of $G$. We can define a partition of $V(G)$ by setting

$$V_1 = \{u \in V(G) : d_G(u, v) \text{ is even}\},$$
$$V_2 = \{u \in V(G) : d_G(u, v) \text{ is odd}\}.$$

It remains to show that $(V_1, V_2)$ is indeed a bipartition of $G$. Suppose that $x$ and $y$ are two vertices of $V_1$, and that $xy \in E(G)$. Let $P$ be a shortest $(x, v)$-path, $Q$ a shortest $(y, v)$-path, and $v_1$ be the first common vertex of $P$ and $Q$. Clearly, since $P$ and $Q$ were shortest paths, their $(v_1, v)$ sections must also be shortest $(v_1, v)$-paths. In particular they are of the same length. Let $P_1$ and $Q_1$ be the $(x, v_1)$ and $(y, v_1)$ sections of $P$ and $Q$, respectively. Then, since $P$ and $Q$ are both of even length, it follows that $P_1$ and $Q_1$ have the same parity. However, this gives rise to the odd cycle $C = P_1Q_1^{-1}x$, and thus the required contradiction. Hence no two vertices of $V_1$ are adjacent. Similarly, no two vertices of $V_2$ are adjacent, and $(V_1, V_2)$ is indeed a bipartition of $G$. □

A similar argument gives the following corollary.

**Corollary 2.1.4** *A connected graph $G$ is bipartite if and only if for every vertex $v$ there is no edge $xy$ with $d(v, x) = d(v, y)$.*

**Proof.** Suppose that $G$ is bipartite with bipartition $(V_1, V_2)$, and let $v$ be a fixed vertex of $G$. Then if $d(v, y) = d(v, y)$ $x$ and $y$ are members of the same colour class and $xy \notin E(G)$.

On the other hand, given a vertex $v \in V$ we define the partition $V_1 = \bigcup_{i\geq 1} N_{2i-1}(v)$ and $V_2 = V(G)\backslash V_1$. Then it follows from our assumptions that $G$ only has edges joining vertices from different $V_i$'s, and therefore that $G$ is bipartite. □

There are many characterisations of bipartite graphs, and therefore many algorithmic ways to recognise them. Corollary 2.1.4 gives rise to one such algorithm: choose a vertex $v \in V(G)$ and consider the level representation of $G$ with respect to $v$. If each $N_i(v)$ spans no edges then $G$ is bipartite, otherwise $G$ is not bipartite. The following variation of Theorem 2.1.3 will be useful later.

**Corollary 2.1.5** *A graph $G$ is bipartite if and only if it contains no closed walk of odd length.*

**Proof.** Since an odd cycle is also an odd walk the condition is certainly sufficient. Thus it suffices to show that a bipartite graph contains no closed walk of odd length. Let $G$ be bipartite and $W = v_0v_1v_2\dots v_kv_0$ be a closed walk in $G$. Consider the level representation of $G$ with respect to $v_0$. We define the sequence $\alpha_1, \alpha_2, \dots \alpha_{k+1}$ by

$$\alpha_i = \begin{cases} 1 & \text{if } 1 \leq i \leq k \text{ and the level of } v_{i-1} \text{ is less than the level of } v_i, \\ 0 & \text{otherwise.} \end{cases}$$

Then, since $W$ is closed, the sequence must contain equal numbers of 1's and 0's, and hence must be of even length. Therefore, $W$ is also of even length. □

# Exercises

2.1.1 Let $V_1 = \{v_1, v_2, \dots, v_m\}$ and $V_2 = \{u_1, \dots, u_n\}$. Show that

(a) △ there are $2^{mn}$ simple bipartite graphs with bipartition $(V_1, V_2)$,

(b) ▽ the number of simple bipartite graphs with even vertex degrees and bipartition $(V_1, V_2)$ is $2^{(m-1)(n-1)}$.

2.1.2 △ Show that a graph $G$ is bipartite if and only if every subgraph $H$ has a set of at least $|V(H)|/2$ mutually non-adjacent vertices.

2.1.3 ▽ Prove that a triangle-free simple graph in which no three vertices have equal degree is bipartite. (Erdős, Fajtlowits, Staton (1991))

2.1.4 ▽ Show that a triangle-free graph $G$ on $p$ vertices with $\delta(G) > 2p/5$ is bipartite. (Andrásfai, Erdős, Sós (1974))

2.1.5 △ Let $G$ be a connected, simple graph with $V(G) = \{v_1, \dots, v_p\}$ and $H$ be a bipartite graph with bipartition $(V_1, V_2)$ where the colour

classes are $V_1=\{x_1,\ldots,x_p\}$, $V_2=\{y_1,\ldots,y_p\}$ and $v_iv_j \in E(G)$ if and only if $x_iy_j$ and $x_jy_i$ are in $E(H)$. Show that $G$ is bipartite if and only if $H$ is not connected.

2.1.6 △ Show that any arbitrary pair of simple graphs $G$ and $H$ are isomorphic if and only if the graphs $bip(G)$ and $bip(H)$ are isomorphic.

## 2.2 Bipartite graphs of certain types

Several special classes of bipartite graphs are of particular interest, and in this section we shall briefly introduce some of these natural special cases.

The first of these is a ***tree*** – a connected, simple graph which contains no cycles; amusingly a union of disjoint trees is called a ***forest***. Trees and forests are certainly bipartite, since they contain no cycles of either parity, and they have an abundance of different characterisations; in the following theorem we give only two, but several more can be found throughout this book.

**Proposition 2.2.1** *The following statements are equivalent for a graph $G$:*

*(1) $G$ is a tree,*

*(2) each pair of vertices is joined by a unique path,*

*(3) $G$ is connected and $|V(G)| = |E(G)| + 1$.*

**Proof.** *(1)* implies *(2)*

Since $G$ is connected every two vertices are connected by some path. Let $u$ and $v$ be two vertices of $G$ which are connected by two distinct $(u,v)$-paths $P_1$ and $P_2$. We shall show that $P_1 \cup P_2$ contains a cycle. Let $w$ be the first vertex of $P_1$ which is also on $P_2$, but whose successor in $P_1$ is not in $P_2$, and let $w'$ be the next vertex of $P_1$ which also lies on $P_2$. Then the segments of $P_1$ and $P_2$ which lie between $w$ and $w'$ together form a cycle.

*(2)* implies *(3)*

Clearly $G$ is connected. Let $p = |V(G)|$ and $q = |E(G)|$. We shall prove the relationship between the numbers of edges and vertices by induction on $p$.

The assertion is clear for connected graphs with one or two vertices. Thus let $p \geq 3$ be a value for which the implication holds for graphs with fewer than $p$ vertices. Delete an edge, $e$, from $G$. Then by assumption $G - e$ will consist of two connected components to which we may apply our induction hypothesis. Each component will have one more vertex than edge and so we must also have $p - 1$ edges in $G$.

*(3)* implies *(1)*

Once again $G$ is connected, and we need only show that $G$ has no cycle. We shall again prove this by induction on the number of vertices. The