Chapter 2
Basic Graph Terminologies

In this chapter, we learn some definitions of basic graph theoretic terminologies and know some preliminary results of graph theory.

2.1 Graphs and Multigraphs

A *graph* G is a tuple consisting of a finite set V of vertices and a finite set E of edges where each edge is an unordered pair of vertices. The two vertices associated with an edge e are called the *end-vertices* of e. We often denote by (u, v), an edge between two vertices u and v. We also denote the set of vertices of a graph G by $V(G)$ and the set of edges of G by $E(G)$. A vertex of a graph is also called as a *node* of a graph.

We generally draw a graph G by representing each vertex of G by a point or a small circle and each edge of G by a line segment or a curve between its two end-vertices. For example, Fig. 2.1 represents a graph G where $V(G) = \{v_1, v_2, \ldots, v_{11}\}$ and $E(G) = \{e_1, e_2, \ldots, e_{17}\}$. We often denote the number of vertices of a graph G by n and the number of edges of G by m; that is, $n = |V(G)|$ and $m = |E(G)|$. We will use these two notations n and m to denote the number of vertices and the number of edges of a graph unless any confusion arises. Thus $n = 11$ and $m = 17$ for the graph in Fig. 2.1.

A *loop* is an edge whose end-vertices are the same. *Multiple edges* are edges with the same pair of end-vertices. If a graph G does not have any loop or multiple edge, then G is called a *simple graph*; otherwise, it is called a *multigraph*. The graph in Fig. 2.1 is a simple graph since it has no loop or multiple edge. On the other hand, the graph in Fig. 2.2 contains a loop e_5 and two sets of multiple edges $\{e_2, e_3, e_4\}$ and $\{e_6, e_7\}$. Hence the graph is a multigraph. In the remainder of the book, when we say a graph, we shall mean a simple graph unless there is any possibility of confusion.

We call a graph a *directed graph* or *digraph* if each edge is associated with a direction, as illustrated in Fig. 2.3(a). One can consider a directed edge as a one-way street. We thus can think an undirected graph as a graph where each edge is directed

M.S. Rahman, *Basic Graph Theory*, Undergraduate Topics in Computer Science,
DOI 10.1007/978-3-319-49475-3_2

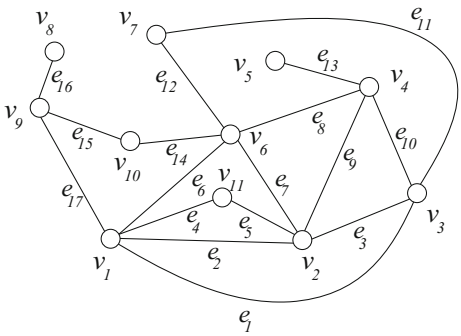

Fig. 2.1 A simple graph with 11 vertices and 17 edges

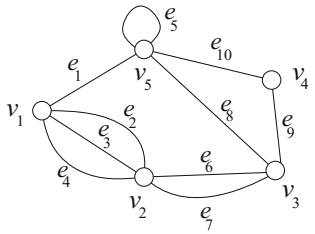

Fig. 2.2 A multigraph

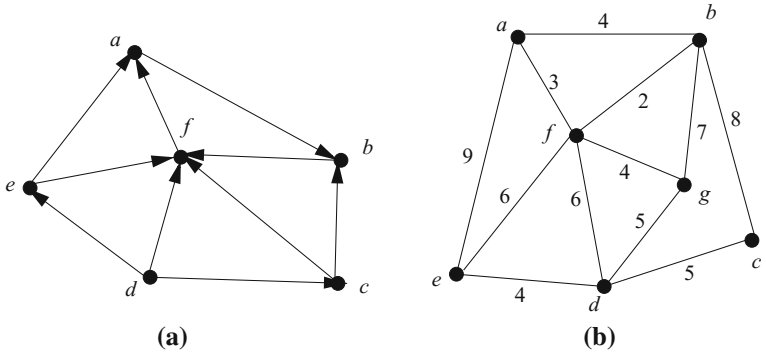

Fig. 2.3 **(a)** A directed graph and **(b)** an edge-weighted graph

in both directions. We deal with digraphs in Chapter 8. We call a graph a *weighted graph* if a weight is assigned to each vertex or to each edge. Figure 2.3(b) illustrates an edge-weighted graph where a weight is assigned to each edge.

2.2 Adjacency, Incidence, and Degree

Let $e = (u, v)$ be an edge of a graph G. Then the two vertices u and v are said to be *adjacent* in G, and the edge e is said to be *incident* to the vertices u and v. The vertex u is also called a *neighbor* of v in G and vice versa. In the graph in Fig. 2.1, the vertices v_1 and v_3 are adjacent; the edge e_1 is incident to the vertices v_1 and v_3. The neighbors of the vertex v_1 in G are v_2, v_3 v_6, v_9, and v_{11}.

The *degree* of a vertex v in a graph G, denoted by $deg(v)$ or $d(v)$, is the number of edges incident to v in G, with each loop at v counted twice. The degree of the vertex v_1 in the graph of Fig. 2.1 is 5. Similarly, the degree of the vertex v_5 in the graph of Fig. 2.2 is also 5.

Since the degree of a vertex counts its incident edges, it is obvious that the summation of the degrees of all the vertices in a graph is related to the total number of edges in the graph. In fact the following lemma, popularly known as the "Degree-sum Formula," indicates that summing up the degrees of each vertex of a graph counts each edge of the graph exactly twice.

Lemma 2.2.1 (Degree-sum Formula) *Let $G = (V, E)$ be a graph with m edges. Then $\sum_{v \in V} deg(v) = 2m$.*

Proof Every nonloop edge is incident to exactly two distinct vertices of G. On the other hand, every loop edge is counted twice in the degree of its incident vertex in G. Thus, every edge, whether it is loop or not, contributes a two to the summation of the degrees of the vertices of G. □

The above lemma, due to Euler (1736), is an essential tool of graph theory and is sometimes refer to as the "First Theorem of Graph Theory" or the "Handshaking Lemma." It implies that if some people shake hands, then the total number of hands shaken must be even since each handshake involves exactly two hands. The following corollary is immediate from the degree-sum formula.

Lemma 2.2.2 *The number of odd degree vertices in a graph is an even number.*

Proof Let G be a graph with m edges. Let x be the sum of the degrees of even degree vertices and y be the sum of the degrees of odd degree vertices. By Lemma 2.2.1 $x + y = 2m$. Since x is the sum of even integers, x is even, and hence $y = 2m - x$ is also an even integer. Since y is the sum of odd integers, the number of addends in the sum must be even. Thus, the number of odd degree vertices must be even. □

2.2.1 Maximum and Minimum Degree

The *maximum degree of a graph* G, denoted by $\Delta(G)$, is the maximum value among the degrees of all the vertices of G, i.e., $\Delta(G) = \max_{v \in V(G)} deg(v)$. Similarly, we define

the *minimum degree of a graph G* and denote it by $\delta(G)$, i.e., $\delta(G) = \min_{v \in V(G)} deg(v)$. The maximum and the minimum degree of the graph in Fig. 2.1 are 5 and 1, respectively.

Let G be a graph with n vertices and m edges. Then by the degree-sum formula, the average degree of a vertex in G is $\frac{2m}{n}$ and hence $\delta(G) \leq \frac{2m}{n} \leq \Delta(G)$.

2.2.2 Regular Graphs

If all the vertices of a graph G have equal degrees, then we call G a *regular graph*. We call it a *k-regular graph* if the common degree is k. Figure 2.4 represents some regular graphs.

Only the graphs with empty edge sets (having nonempty vertex sets) are 0-regular. These are called *null graphs*. Similarly, a 1-regular graph consists of a set of edges such that no two edges are incident to a common vertex. A graph which consists of a list of vertices such that each pair of consecutive vertices are adjacent with each other and the first vertex is also adjacent with the last vertex, as illustrated in Fig. 2.4(a), is a 2-regular graph. Such a graph is called a *cycle* or a *simple cycle*. A graph which is a collection of simple cycles is also a 2-regular graph.

A 3-regular graph is also called a *cubic graph*. The graph in Fig. 2.4(b) is a cubic graph. This particular graph is also known as the "Petersen graph" after the name of Julius Petersen, who constructed it in 1898. This graph has 10 vertices and 15 edges. The vertices of Petersen graph can be labeled by two element subsets of the set $\{1, 2, 3, 4, 5\}$ such that the labels of two adjacent vertices are disjoint, as illustrated in Fig. 2.5. Petersen graph shows some interesting properties and also serves as a minimum-sized example and counterexample for many problems in graph theory. Doughnut graphs [1] are examples of 5-regular graphs. A p-doughnut graph has exactly $4p$ vertices. Figure 2.4(d) illustrates a p-doughnut graph for $p = 4$. Another important example of a regular graph is a "d-dimensional hypercube" or simply "hypercube." A d-dimensional hypercube has 2^d vertices and each of its vertices has degree d. Figure 2.4(c) illustrates a d-dimensional hypercube for $d = 4$.

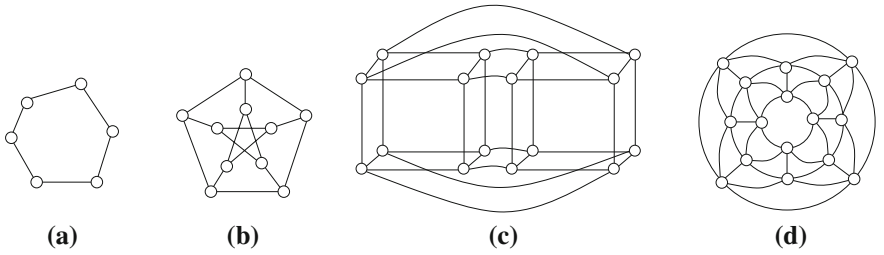

Fig. 2.4 **(a)** A 2-regular graph (a simple cycle), **(b)** a 3-regular graph (Petersen graph), **(c)** a 4-regular graph (4-dimensional hypercube), and **(d)** a 5-regular graph (a doughnut graph)

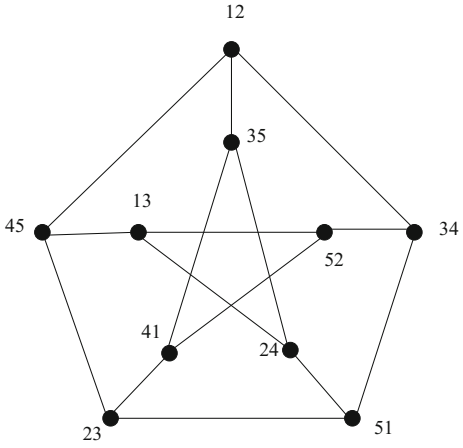

Fig. 2.5 A labeled Petersen graph

The degree-sum formula implies the following two corollaries for regular graphs.

Corollary 2.2.3 *Every regular graph with an odd degree has an even number of vertices.*

Corollary 2.2.4 *A k-regular graph with n vertices has $nk/2$ edges.*

2.3 Subgraphs

A subgraph of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. For instance, the graphs in Figs. 2.6(b)–(e) are subgraphs of the graph in Fig. 2.6(a).

We can obtain subgraphs of a graph G by deleting some vertices and edges of G. Let e be an edge of G. We denote by $G - e$ the graph obtained by deleting the edge e from G. More generally, if F is a set of edges of G, we denote by $G - F$ the graph obtained by deleting all the edges in F from G. Figure 2.6(a) illustrates a graph G and Fig. 2.6(b) illustrates the graph $G - e_6$ obtained by deleting the edge e_6 from G.

Similarly, we can define the deletion of a vertex from a graph. However, deleting a vertex v from a graph G also requires that we also delete the edges incident to v in G. Let v be a vertex of a graph G. We denote by $G - v$ the graph obtained by deleting the vertex v and all its incident edges from G. More generally, if W is a set of vertices of G, we denote by $G - W$ the graph obtained by deleting the vertices in W (and all the incident edges) from G. Figure 2.6(a) illustrates a graph G and Fig. 2.6(c) illustrates the graph $G - v_7$ obtained by deleting the vertex v_7 from G.

Let $G = (V, E)$ be a graph and let W be a set of vertices of G. A subgraph $G' = (V', E')$ of G is called *a subgraph of G induced by W* if $V' = W$ and E'

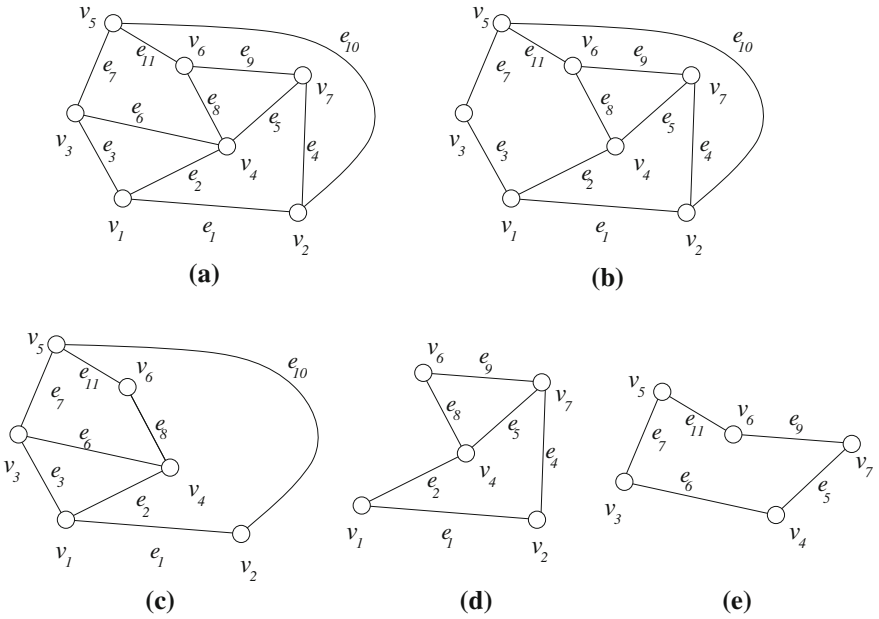

Fig. 2.6 **(a)** A Graph G, **(b)–(e)** subgraphs of G

consists of all those edges e of G such that both the end-vertices of e are in W. The graph in Fig. 2.6(d) is a subgraph of the graph of Fig. 2.6(a) induced by the set of vertices $\{v_1, v_2, v_4, v_6, v_7\}$.

Let $G = (V, E)$ be a graph and let F be a set of edges of G. A subgraph $G' = (V', E')$ of G is called *a subgraph of G induced by F* if $E' = F$ and V' consists of all those vertices of G each of which is an end-vertex of some edge in F. The graph in Fig. 2.6(e) is a subgraph of the graph of Fig. 2.6(a) induced by the set of edges $\{e_5, e_6, e_7, e_9, e_{11}\}$.

2.4 Some Important Trivial Classes of Graphs

In this section, we see some special classes of graphs, which will often appear in our discussion in the subsequent chapters.

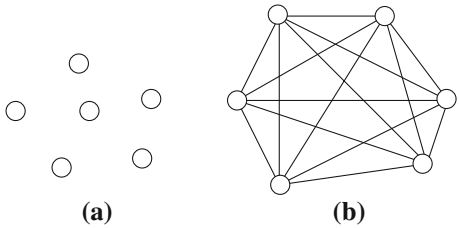

Fig. 2.7 **(a)** A null graph N_6 with six vertices, **(b)** a complete graph K_6 with six vertices

2.4.1 Null Graphs

A graph with an empty edge set is called a *null graph*. A null graph with n vertices is denoted by N_n. Figure 2.7(a) illustrates the null graph N_6 with six vertices. A null graph is a subgraph of any graph with the same number of vertices.

2.4.2 Complete Graphs

A graph in which each pair of distinct vertices are adjacent is called a *complete graph*. A complete graph with n vertices is denoted by K_n. It is trivial to see that K_n contains $n(n-1)/2$ edges. Figure 2.7(b) illustrates a complete graph K_6 with six vertices. Any graph is a subgraph of the complete graph with the same number of vertices and thus the number of edges in a graph with n vertices is at most $n(n-1)/2$.

2.4.3 Independent Set and Bipartite Graphs

Let $G = (V, E)$ be a graph. A subset of vertices $V' \subseteq V$ is called an *independent set* in G if for every pair of vertices $u, v \in V'$, there is no edge in G joining the two vertices u and v.

A graph G is called a *bipartite graph* if the vertex set V of G can be partitioned into two disjoint nonempty sets V_1 and V_2, both of which are independent. The two sets V_1 and V_2 are often called the *partite sets* of G. Each edge of a bipartite graph G thus joins exactly one vertex of V_1 to exactly one vertex of V_2. Figure 2.8 shows two bipartite graphs where the independent partitions are shaded in both the graphs. Given a graph G, one can test whether G is a bipartite graph in a naive approach by considering each possible bipartition of the vertices of G and checking whether the two partitions are independent or not. However since there are $2^n - 2$ possible bipartition of a graph with n vertices, this approach takes exponential time. Fortunately, there is a linear-time algorithm to test whether a graph is bipartite or not. The idea is simple. Using a breadth first search (BFS) on the graph G, color

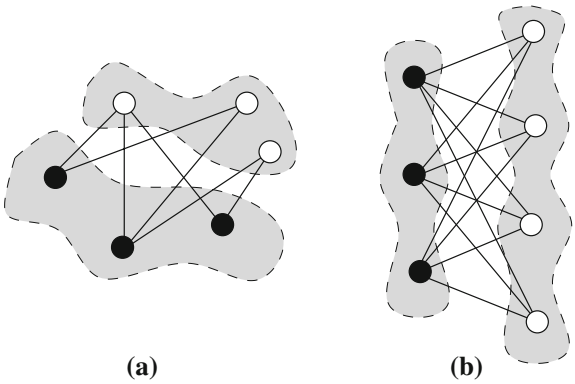

Fig. 2.8 Two bipartite graphs: the two independent partite sets are highlighted for each of them, one containing the *black*-colored vertices and the other containing the *white*-colored vertices

the vertices of G with two colors so that no two adjacent vertices receive the same color. We say colors of two vertices *conflict* if the vertices are adjacent and receive the same color. If a conflict-free coloring can be done by BFS, then G is bipartite, otherwise not.

Let G be a bipartite graph with the two independent sets V_1 and V_2. We call G a *complete bipartite graph* if for each vertex $u \in V_1$ and each vertex $v \in V_2$, there is an edge (u, v) in G. Figure 2.8(b) illustrates a complete bipartite graph where the two partite sets contains 3 and 4 vertices, respectively. This graph is denoted by $K_{3,4}$. In general, a complete bipartite graph is denoted by $K_{m,n}$ if its two partite sets contain m and n vertices, respectively. One can easily see that $K_{m,n}$ contains $m \times n$ edges.

2.4.4 Path Graphs

A *path graph* is a graph G that contains a list of vertices $v_1, v_2, \ldots, v_p$ of G such that for $1 \leq i \leq p-1$, there is an edge (v_i, v_{i+1}) in G and these are the only edges in G. The two vertices v_1 and v_p are called the *end-vertices* of G. Figure 2.9(a) illustrates a path graph with six vertices. A path graph with n vertices is denoted by P_n. Note that the degree of each vertex of a path graph is two except for the two end-vertices, both of which have degree one.

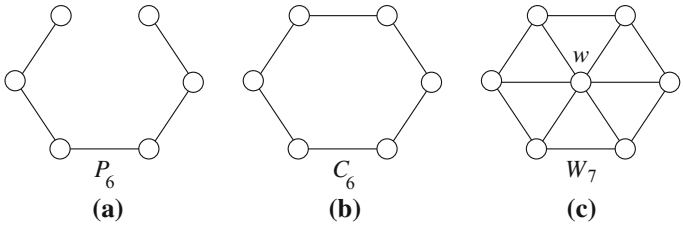

Fig. 2.9 **(a)** P_6, **(b)** C_6, and **(c)** W_7

2.4.5 Cycle Graphs

A *cycle graph* is one that is obtained by joining the two end-vertices of a path graph. Thus, the degree of each vertex of a cycle graph is two. Figure 2.9(b) illustrates a cycle graph with six vertices. A cycle graph with n vertices is often denoted by C_n.

2.4.6 Wheel Graphs

A *wheel graph* with n vertices, denoted by W_n, is obtained from a cycle graph C_{n-1} with $n-1$ vertices by adding a new vertex w and joining an edge from w to each vertex of C_{n-1}. Figure 2.9(c) illustrates a wheel graph with seven vertices.

2.5 Operations on Graphs

We are already familiar with two operations on graphs, namely deletion of vertices and deletion of edges. In this section, we see some other operations on graphs. Since a graph $G = (V, E)$ is defined as a tuple of two sets; the vertex set and the edge set, some operations on sets can naturally be extended to graphs. We first show some examples of such set operations on graphs. Later in this section, we also define some other operations on graphs.

2.5.1 Union and Intersection of Graphs

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. The *union* of G_1 and G_2, denoted by $G_1 \cup G_2$, is another graph $G_3 = (V_3, E_3)$, whose vertex set $V_3 = V_1 \cup V_2$ and edge set $E_3 = E_1 \cup E_2$.

Similarly, the *intersection* of G_1 and G_2, denoted by $G_1 \cap G_2$, is another graph $G_4 = (V_4, E_4)$, whose vertex set $V_4 = V_1 \cap V_2$ and edge set $E_4 = E_1 \cap E_2$.

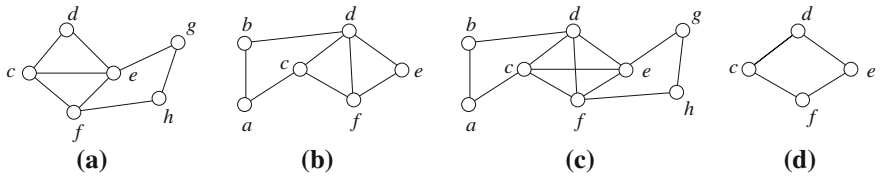

Fig. 2.10 **(a)** G_1, **(b)** G_2, **(c)** $G_1 \cup G_2$, and **(d)** $G_1 \cap G_2$

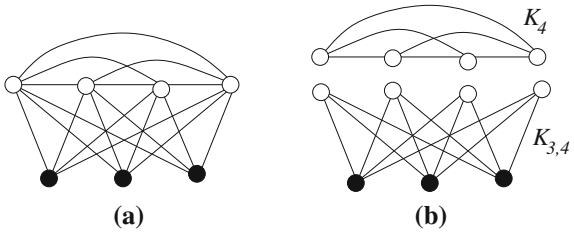

Fig. 2.11 **(a)** The graph representing handshakes, **(b)** K_4 and $K_{3,4}$

Figures 2.10(a) and (b) show two graphs G_1 and G_2, and Figs. 2.10(c) and (d) illustrate their union and intersection, respectively.

Clearly, we can define the union and intersection of more than two graphs in a similar way. These operations on graphs can be used to solve many problems very easily. We now present such an application of these operations on two graphs [2].

Suppose there are $h + g$ people in a party; h of them are hosts and g of them are guests. Each person shakes hands with each other except that no host shakes hands with any other host. The problem is to find the total number of handshakes. As usual, we transform the scenario into a graph problem as follows. We form a graph with $h + g$ vertices; h of them are black vertices, representing the hosts and the other g vertices are white, representing the guests. The edges of the graph represent the handshakes. Thus, there is an edge between every pair of vertices except for that there is no edge between any pair of black vertices. Thus, the problem now is to count the number of edges in the graph thus formed. The graph is illustrated for $h = 3$ and $g = 4$ in Fig. 2.11(a).

To solve the problem, we note that the graph can be thought of as a union of two graphs: a complete graph K_g and a complete bipartite graph $K_{h,g}$ as illustrated in Fig. 2.11(b). Since there is no common edge between the two graphs, their intersection contains no edges. Thus, the total number of edges in the graph (i.e., the total number of handshakes in the party) is $n(n-1)/2 + m \times n$.

2.5.2 Complement of a Graph

The *complement* of a graph $G = (V, E)$ is another graph $\overline{G} = (V, \overline{E})$ with the same vertex set such that for any pair of distinct vertices $u, v \in V$, $(u, v) \in \overline{E}$

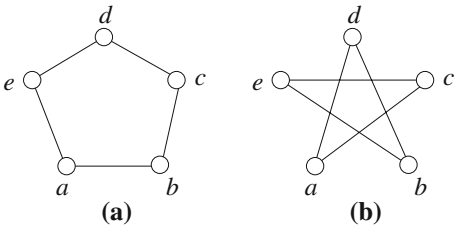

Fig. 2.12 **(a)** A graph G, and **(b)** the complement $\overline{G}$ of G

if and only if $(u, v) \notin E$. We often denote the complement of a graph G by $\overline{G}$. Figure 2.12(b) illustrates the complement of the graph in Fig. 2.12(a). A null graph is the complement of the complete graph with the same number of vertices and vice versa. The following lemma is an interesting observation in terms of the complement of a graph.

Lemma 2.5.1 *For any graph of six vertices, G or $\overline{G}$ contains a triangle.*

Proof Let G be a graph of six vertices, and let v be a vertex of G. Since the total number of neighbors of v in G and $\overline{G}$ is five, v has at least three neighbors either in G or in $\overline{G}$ by the pigeonhole principle. Without loss of generality we can assume that v has three neighbors x, y and z in G. If any two of x, y, and z are adjacent to each other, then G contains a triangle. If no two of x, y, and z are adjacent, then x, y, and z will form a triangle in $\overline{G}$. □

2.5.3 Subdivisions

Subdividing an edge (u, v) of a graph G is the operation of deleting the edge (u, v) and adding the path u, w, v through a new vertex w of degree two. A graph G' is said to a *subdivision of a graph* G if G' can be obtained from G by successively subdividing some of the edges of G. Figure 2.13(b) illustrates a subdivision of the graph in Fig. 2.13(a).

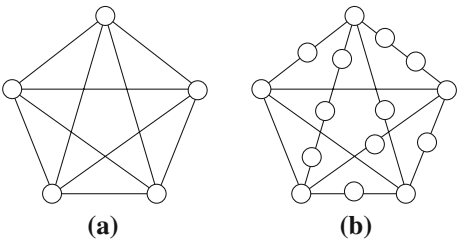

Fig. 2.13 **(a)** A graph G, and **(b)** a subdivision G' of G

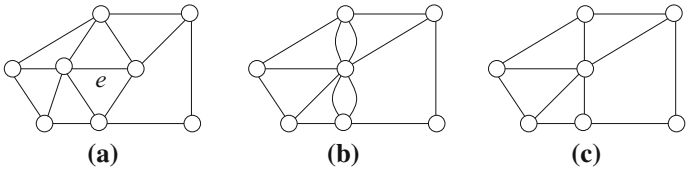

Fig. 2.14 **(a)** A graph G, **(b)** the graph obtained by contracting the edge e in G, and **(c)** the simple graph obtained by contracting the edge e in G

2.5.4 Contraction of an Edge

The *contraction* of an edge (u, v) of a graph G is the operation of deleting the edge (u, v) and identifying the two vertices u and v. Thus to contract the edge (u, v), we delete the two vertices u, v and add a new vertex w where all the edges incident to u and v in G other than the edge (u, v) are made incident to w. Figure 2.14(a) illustrates a graph G and Fig. 2.14(b) shows the new graph obtained by contracting the edge e in G. We denote by $G\backslash e$ the graph obtained from G by contracting an edge e.

After contracting an edge (u, v) of a graph, the new graph may contain multi-edges. For example, the graph in Fig. 2.14(b) contains some multi-edges. When we require only a simple graph for our consideration, then we often take a simple graph by replacing each set of multi-edges by a single edge from the multigraph. For example Fig. 2.14(c) illustrates a simple graph obtained by contracting the edge e of the graph in Fig. 2.14(a).

2.6 Graph Isomorphism

An *isomorphism* between two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is a one-to-one correspondence between the vertices in V_1 and V_2 such that the number of edges between any two vertices in V_1 is equal to the number of edges between the corresponding two vertices in V_2. Thus, an *isomorphism* between two simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is defined to be a bijection $f : V_1 \longrightarrow V_2$ such that for any two vertices u and v of G_1, $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$. If there is an isomorphism between two graphs G_1 and G_2, then we say that G_1 is *isomorphic* to G_2 and write $G_1 \cong G_2$.

Figure 2.15(a) shows two graphs G_1 and G_2 that are isomorphic to each other. This isomorphism can be noticed by mapping the vertices a, b, c, d, and e of G_1 to the vertices u, w, y, v, and x of G_2, respectively. Similarly, the two graphs illustrated in Fig. 2.15(b) are also isomorphic. This isomorphic class of graphs is known as Petersen Graph. Observe the two graphs in Fig. 2.15(a). If we map the vertices a, b, c, d, and e of G_1 to the vertices u, v, w, x, and y of G_2, respectively, then the two graphs become complement to each other. A graph which is isomorphic to

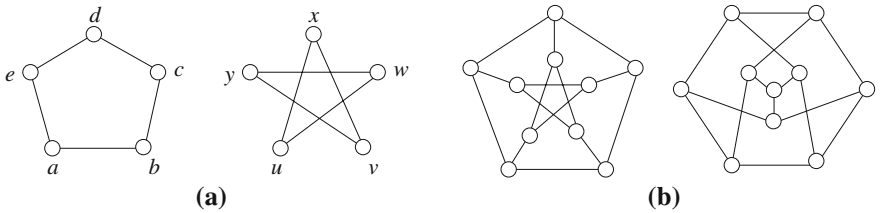

Fig. 2.15 Two pairs of isomorphic graphs

its complement is called a *self-complimentary*. The graphs in Fig. 2.15(a) (C_5) are self-complimentary. Similarly, P_4 is also self-complimentary.

In order to decide whether two graphs G_1 and G_2 are isomorphic or not, a trivial approach would take all the possible permutations of the vertices of G_2 to check whether any of these permutations induces an isomorphism. Clearly, this approach takes exponential time on the number of vertices. Unfortunately, there is no known polynomial-time algorithm to examine whether two graphs are isomorphic or not; neither there is any proof of the claim that there cannot be any such polynomial-time algorithm. Thus, it is an interesting open problem to prove whether the existence or inexistence of a polynomial-time algorithm to test two graphs to be isomorphic.

We end this section with the following lemma, which shows that isomorphism between graphs gives equivalence classes under the isomorphism relation [3].

Lemma 2.6.1 *The isomorphism relation is an equivalence relation on the set of graphs.*

Proof The reflexivity property of the isomorphism is trivial since for any graph $G = (V, E)$, the bijection $f : V \longrightarrow V$ that maps every vertex $v \in V$ to itself gives an isomorphism. Again if $f : V_1 \longrightarrow V_2$ is an isomorphism from a graph $G_1 = (V_1, E_1)$ to another graph $G_2 = (V_2, E_2)$, then f^{-1} defines an isomorphism from G_2 to G_1, because $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$ implies that $(x, y) \in E_2$ if an only if $(f^{-1}(x), f^{-1}(y)) \in E_1$. Thus, isomorphism relation is also symmetric on the set of graphs. We now prove the transitivity property of isomorphism. Suppose $f : V_1 \longrightarrow V_2$ and $g : V_2 \longrightarrow V_3$ define isomorphisms from G_1 to G_2 and from G_2 to G_3, where $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ and $G_3 = (V_3, E_3)$ are three graphs. Hence, $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$ and $(x, y) \in E_2$ if and only if $(g(x), g(y)) \in E_3$. Therefore, $(u, v) \in E_1$ if and only if $(g(f(u)), g(f(v))) \in E_3$. Thus $g \circ f$ defines an isomorphism from G_1 to G_3. □

For many problems, we often require only structural properties of a graph. For such cases, the "labels" of the vertices of the graph are often unnecessary, and we informally use the term "unlabeled graphs" to denote an isomorphic class of graphs.

2.7 Degree Sequence

The *degree sequence* of a graph is the list of vertex degrees. The degree sequence of a graph is usually written in nonincreasing order as $d_1 \geq \cdots \geq d_n$. The set of distinct nonnegative integers occurring in a degree sequence of a graph is called its *degree set.* For the graph in Fig. 2.16(a) the degree sequence is 5, 4, 3, 3, 3, 2 and the degree set is {2, 3, 4, 5}. Two graphs with the same degree sequence are said to be *degree equivalent.* The two graphs in Figs. 2.16(a) and (b) are degree equivalent.

Every graph has a degree sequence. But does there exist a graph for a given sequence of nonincreasing nonnegative integers? To answer this question, a trivial necessary condition comes from degree-sum formula. That is, the sum of the integers in the sequence must be even. It is interesting that this trivial necessary condition is also sufficient, as can be seen from the following construction. Assume that the sum of the integers in the sequence is even. Since the sum is even, the number of odd values in the sequence is even. First form an arbitrary pairing of the vertices with odd degrees, and add an edge between the two vertices of each pair. Then the remaining degree needed to each vertex is even and nonnegative. Construct loops in each vertex to fulfill these degree requirements of each vertex. The construction of a graph for the sequence 5, 4, 3, 2, 1, 1 is illustrated in Fig. 2.17.

Allowing loops makes the realization of a degree sequence easy. If we do not allow loops and multiple edges, some sequences are not realizable even if their sum is even. For example, it is not possible to realize the sequence 2, 2, 0 if we do not allow loops and multiple edges. We call a degree sequence a *graphic sequence* if it realizes a simple graph. Thus, the degree sequence 2, 2, 0 is not a graphic sequence. Figure 2.18(a) shows a simple graph G with the graphic sequence 3, 3, 3, 2, 1 with $\Delta = 3$. We can construct a simple graph G' from G by adding a new vertex x and adding edges x to each of the $\Delta + 1$ vertices of largest degrees in G, as illustrated in Fig. 2.18(b), where the degree sequence of G' is 4, 4, 4, 4, 3, 1. Now consider the

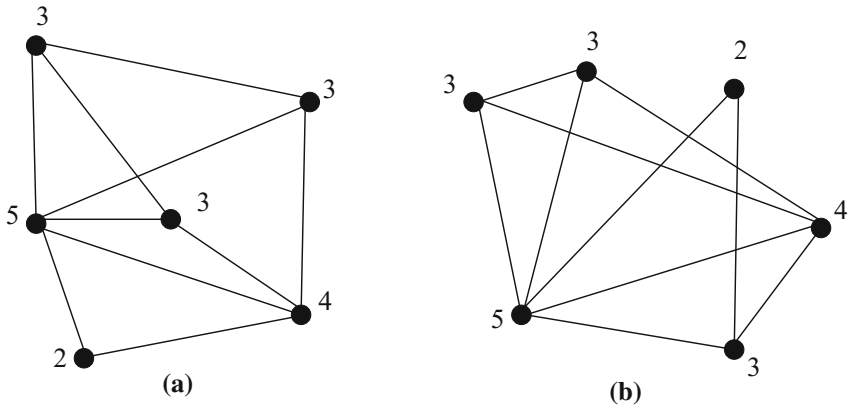

Fig. 2.16 Illustration for degree sequence

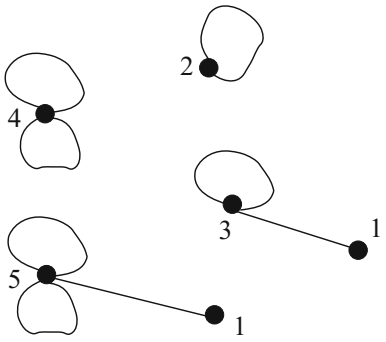

Fig. 2.17 Illustration of a construction for the sequence 5, 4, 3, 2, 1, 1

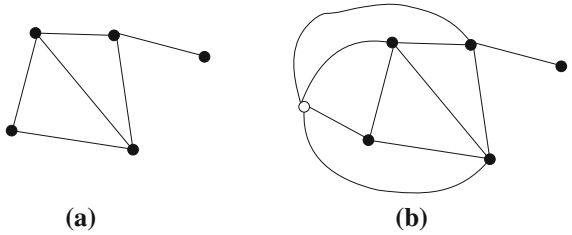

Fig. 2.18 **(a)** A graph G with the degree sequence 3, 3, 3, 2, 1 and **(b)** a graph G' with the degree sequence 4, 4, 4, 4, 3, 1

following scenario. Assume that we have a degree sequence $d = d_1, d_2, \ldots, d_n$. We obtain a degree sequence d' from d by deleting d_1 and subtracting 1 from each of $d_2, \ldots, d_{d_1+1}$. Then clearly d is graphic if d' is graphic, since we can construct a simple graph realizing d from a simple graph realizing d' by adding a new vertex and d_1 edges. The reverse of the implication above is also hold as Havel [4] and Hakimi [5] showed that d is graphic if and only if d' is graphic. Therefore, based on the construction above, we can easily develop a recursive algorithm to check whether a degree sequence is graphic or not. Note that $d_1 = 0$ is the only 1-element graphic sequence. Steps of an recursive algorithm for testing a graphic sequence are illustrated in Fig. 2.19. Note that when we obtain d' from d, we rearrange d' in nonincreasing

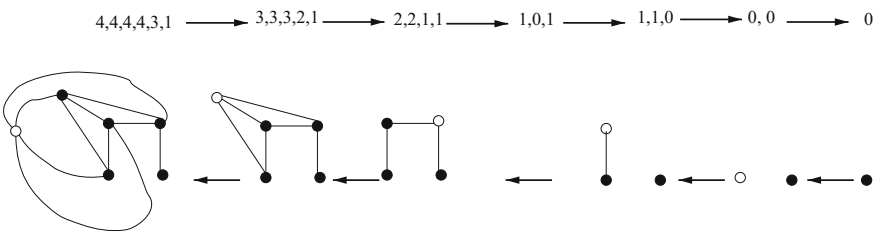

Fig. 2.19 Illustration for testing the realization of a graphic sequence

order if d' is not in nonincreasing order. For example, observe $1, 0, 1 \rightarrow 1, 1, 0$ in Fig. 2.19.

2.8 Data Structures and Graph Representation

We are already accustomed with one way of representing a graph, i.e., by a diagram where each vertex is represented by a point or a small circle and each edge is represented by a straight-line between the end-vertices. This graphical representation is convenient for the visualization of a graph, yet it is unsuitable if we want to store a graph in a computer. However, there are other ways of representing a graph. In this section, we first give a very brief account of some basic data structures and then we show three methods for representing a graph in a computer.

A vector or a set of variables is usually stored as a (one-dimensional) array and a matrix is stored as a two-dimensional array. The main feature of an array is that the location of an entry can be uniquely determined by its index in the array and the entry can be accessed in constant time. A list is a data structure which consists of homogeneous records, linked together in a linear fashion. Each record contains one or more items of data and one or more pointers. In a *singly linked lists*, each record has a single forwarding pointer indicating the address of the memory cell of the next record. In a *doubly linked list*, each record has forward and backward pointers indicating the address of the memory cells of the next and the previous records, respectively.

We now present three different representations of graphs.

2.8.1 Adjacency Matrix

Let G be a graph with the vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and the edge set $E = \{e_1, e_2, \ldots, e_m\}$. The adjacency matrix $A(G)$ of G is an $n \times n$ matrix $A(G) = [a_{ij}]$ in which a_{ij} is the number of edges between the two vertices v_i and v_j. Figure 2.20(a) shows a graph and Fig. 2.20(b) illustrates its adjacency matrix. Note that an adjacency matrix of a graph is always symmetric.

If G is a simple graph, then each entry of $A(G)$ is either a zero or one and the main diagonal of the matrix contains only zeros. Figure 2.21(b) illustrates the adjacency matrix of the simple graph in Fig. 2.21(a).

An adjacency matrix uses $O(n^2)$ space to represent a graph of n vertices. It is not economical when the number of edges in the graph is much less than the maximum possible $n(n-1)/2$. If $A(G)$ is stored as a two-dimensional array, then checking whether $(v_i, v_j) \in E$ for a pair of vertices v_i and v_j or deleting an edge (v_i, v_j) from G requires only constant time. However, scanning all the neighbors of a vertex v requires n steps even if $deg(v)$ is much less than n.

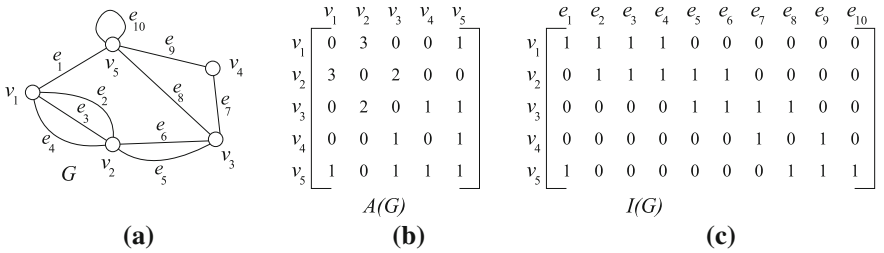

Fig. 2.20 **(a)** A graph G, **(b)** adjacency matrix representation of G, and **(c)** incidence matrix representation of G

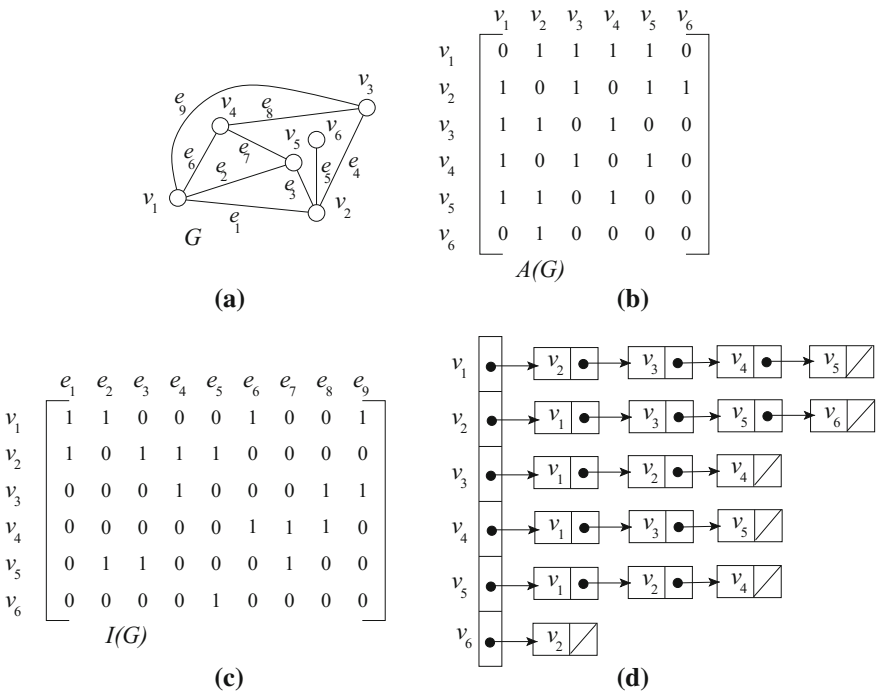

Fig. 2.21 **(a)** A simple graph G, **(b)** adjacency matrix representation of G, **(c)** incidence matrix representation of G, and **(d)** adjacency list representation G

2.8.2 *Incidence Matrix*

Let G be a graph with the vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and the edge set $E = \{e_1, e_2, \ldots, e_m\}$. The incidence matrix $I(G)$ of G is an $n \times m$ matrix $M(G) = [m_{ij}]$ in which m_{ij} is 1 if the vertex v_i is adjacent to the edge e_j, and 0 otherwise. Figure 2.20(c) illustrates the incidence matrix of the graph in Fig. 2.20(a). Similarly, Fig. 2.21(c) illustrates the incidence matrix of the graph in Fig. 2.21(a).

The space requirement for the incidence matrix is $n \times m$. For a graph which contains much more number of edges compared to n, this requirement is much higher than the adjacency matrix. Scanning all the neighbors of the graph also takes n steps even if $deg(v)$ is much less than n. However, this representation is helpful for the query to know whether a vertex is incident to an edge or not and this query can be responded in constant time from this representation.

If a graph G is a simple graph, then there is another representation of G, called the adjacency lists.

2.8.3 Adjacency List

Let G be a graph with the vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and the edge set $E = \{e_1, e_2, \ldots, e_m\}$. The adjacency lists $Adj(G)$ of G is an array of n lists, where for each vertex v of G, there is a list corresponding to v, which contains a record for each neighbor of v. Figure 2.21(d) illustrates the adjacency lists of the graph in Fig. 2.21(a).

The space requirement for the adjacency lists is $\sum_{v \in V}(1 + deg(v)) = O(n + m)$. Thus, this representation is much more economical than the adjacency matrix and the incidence matrix, particularly if the number of edge in the graph is much less than $n(n-1)/2$. Scanning the neighbors of a vertex v takes $O(deg(v))$ steps only but checking whether $(v_i, v_j) \in E$ requires $O(deg(v_i))$ steps for a pair of vertices v_i and v_j of G.

Bibliographic Notes

The books [3, 6–9] were used in preparing this chapter.

Exercises

1. Show that every regular graph with an odd degree has an even number of vertices.
2. Construct the complement of $K_{3,3}$, W_5, and C_5.
3. Can you construct a disconnected graph G of two or more vertices such that $\overline{G}$ is also disconnected. Give a proof supporting your answer.
4. Give two examples of self-complementary graphs.
5. What is the necessary and sufficient condition for $K_{m,n}$ to be a regular graph?
6. Is there a simple graph of n vertices such that the vertices all have distinct degrees? Give a proof supporting your answer.
7. Draw the graph $G = (V, E)$ with vertex set $V = \{a, b, c, d, e, f, g, h\}$ and edge set $\{(a, b), (a, e), (b, c), (b, d), (c, d), (c, g), (d, e)(e, f), (f, g), (f, h), (g, h)\}$. Draw $G - (d, e)$. Draw the subgraph of G induced by $\{c, d, e, f\}$. Contract the edge (d, e) from G.
8. Show that two graphs are isomorphic if and only if their complements are isomorphic.

References

1. Karim, M.R., Rahman, M.S.: On a class of planar graphs with straight-line grid drawings on linear area. J. Graph Algorithms Appl. **13**(2), 153–177 (2009)
2. Koshy, T.: Discrete Mathematics with Applications. Elsevier, Amsterdam (2004)
3. West, D.B.: Introduction to Graph Theory, 2nd edn. Prentice Hall, New Jersey (2001)
4. Havel, V.: A remark on the existence of finite graphs [Czec] Casopis. Pest. Mat. **80**, 477–480 (1955)
5. Hakimi, S.L.: On realizability of a set of integers as degrees of vertices of a linear graph. SIAM J. Appl. Math. **10**, 496–506 (1962)
6. Nishizeki, T., Rahman, M.S.: Planar graph drawing. World Scientific, Singapore (2004)
7. Wilson, R.J.: Introduction to Graph Theory, 4th edn. Longman, London (1996)
8. Clark, J., Holton, D.A.: A First Look at Graph Theory. World Scientific, Singapore (1991)
9. Pirzada, S.: An Introduction to Graph Theory. University Press, India (2009)

http://www.springer.com/978-3-319-49474-6

Basic Graph Theory

Rahman, M.S.

2017, X, 169 p. 147 illus., Softcover
ISBN: 978-3-319-49474-6