10.2.5 Bipartite Graphs and Matchings

Bipartite graphs can be used to model many types of applications that involve matching the elements of one set to elements of another, as Example 14 illustrates.

EXAMPLE 14 **Job Assignments** Suppose that there are m employees in a group and n different jobs that need to be done, where $m \geq n$. Each employee is trained to do one or more of these n jobs. We would like to assign an employee to each job. To help with this task, we can use a graph to model employee capabilities. We represent each employee by a vertex and each job by a vertex. For each employee, we include an edge from that employee to all jobs that the employee has been trained to do. Note that the vertex set of this graph can be partitioned into two disjoint sets, the set of employees and the set of jobs, and each edge connects an employee to a job. Consequently, this graph is bipartite, where the bipartition is (E, J) where E is the set of employees and J is the set of jobs. We now consider two different scenarios.

First, suppose that a group has four employees: Alvarez, Berkowitz, Chen, and Davis; and suppose that four jobs need to be done to complete Project 1: requirements, architecture, implementation, and testing. Suppose that Alvarez has been trained to do requirements and testing; Berkowitz has been trained to do architecture, implementation, and testing; Chen has been trained to do requirements, architecture, and implementation; and Davis has only been trained to do requirements. We model these employee capabilities using the bipartite graph in Figure 10(a).

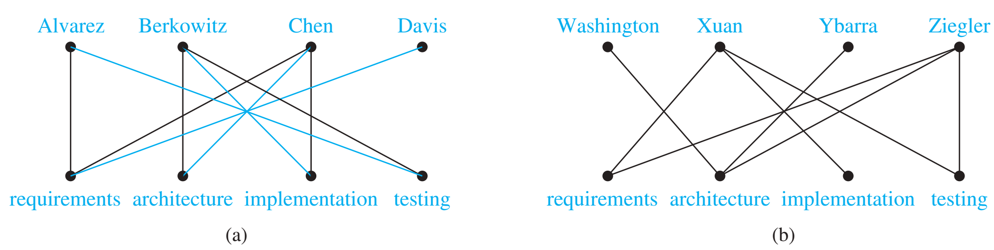


FIGURE 10 **Modeling the jobs for which employees have been trained.**

Second, suppose that a second group also has four employees: Washington, Xuan, Ybarra, and Ziegler; and suppose that the same four jobs need to be done to complete Project 2 as are needed to complete Project 1. Suppose that Washington has been trained to do architecture; Xuan has been trained to do requirements, implementation, and testing; Ybarra has been trained to do architecture; and Ziegler has been trained to do requirements, architecture and testing. We model these employee capabilities using the bipartite graph in Figure 10(b).

To complete Project 1, we must assign an employee to each job so that every job has an employee assigned to it, and so that no employee is assigned more than one job. We can do this by assigning Alvarez to testing, Berkowitz to implementation, Chen to architecture, and Davis to requirements, as shown in Figure 10(a) (where blue lines show this assignment of jobs).

To complete Project 2, we must also assign an employee to each job so that every job has an employee assigned to it and no employee is assigned more than one job. However, this is impossible because there are only two employees, Xuan and Ziegler, who have been trained for at least one of the three jobs of requirements, implementation, and testing. Consequently, there is no way to assign three different employees to these three jobs so that each job is assigned an employee with the appropriate training. ◀

Finding an assignment of jobs to employees can be thought of as finding a matching in the graph model, where a **matching** M in a simple graph $G = (V, E)$ is a subset of the set E of edges of the graph such that no two edges are incident with the same vertex. In other words, a matching is a subset of edges such that if $\{s, t\}$ and $\{u, v\}$ are distinct edges of the matching, then s, t, u, and v are distinct. A vertex that is the endpoint of an edge of a matching M is said to be **matched** in M; otherwise it is said to be **unmatched**. A **maximum matching** is a matching with the largest number of edges. We say that a matching M in a bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) is a **complete matching from** V_1 **to** V_2 if every vertex in V_1 is the endpoint of an edge in the matching, or equivalently, if $|M| = |V_1|$. For example, to assign jobs to employees so that the largest number of jobs are assigned employees, we seek a maximum matching in the graph that models employee capabilities. To assign employees to all jobs we seek a complete matching from the set of jobs to the set of employees. In Example 14, we found a complete matching from the set of jobs to the set of employees for Project 1, and this matching is a maximum matching, and we showed that no complete matching exists from the set of jobs to the employees for Project 2.

We now give an example of how matchings can be used to model marriages.

EXAMPLE 15 **Marriages on an Island** Suppose that there are m men and n women on an island. Each person has a list of members of the opposite gender acceptable as a spouse. We construct a bipartite graph $G = (V_1, V_2)$ where V_1 is the set of men and V_2 is the set of women so that there is an edge between a man and a woman if they find each other acceptable as a spouse. A matching in this graph consists of a set of edges, where each pair of endpoints of an edge is a husband-wife pair. A maximum matching is a largest possible set of married couples, and a complete matching of V_1 is a set of married couples where every man is married, but possibly not all women. ◀

NECESSARY AND SUFFICIENT CONDITIONS FOR COMPLETE MATCHINGS We now turn our attention to the question of determining whether a complete matching from V_1 to V_2 exists when (V_1, V_2) is a bipartition of a bipartite graph $G = (V, E)$. We will introduce a theorem that provides a set of necessary and sufficient conditions for the existence of a complete matching. This theorem was proved by Philip Hall in 1935.

Hall's marriage theorem is an example of a theorem where obvious necessary conditions are sufficient too.

THEOREM 5 **HALL'S MARRIAGE THEOREM** The bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subsets A of V_1.

Proof: We first prove the *only if* part of the theorem. To do so, suppose that there is a complete matching M from V_1 to V_2. Then, if $A \subseteq V_1$, for every vertex $v \in A$, there is an edge in M connecting v to a vertex in V_2. Consequently, there are at least as many vertices in V_2 that are neighbors of vertices in V_1 as there are vertices in V_1. It follows that $|N(A)| \geq |A|$.

Links

Courtesy of the Edinburgh Mathematical Society

PHILIP HALL (1904–1982) Philip Hall grew up in London, where his mother was a dressmaker. He won a scholarship for board school reserved for needy children, and later a scholarship to King's College of Cambridge University. He received his bachelors degree in 1925. In 1926, unsure of his career goals, he took a civil service exam, but decided to continue his studies at Cambridge after failing.

In 1927 Hall was elected to a fellowship at King's College; soon after, he made his first important discovery in group theory. The results he proved are now known as Hall's theorems. In 1933 he was appointed as a Lecturer at Cambridge, where he remained until 1941. During World War II he worked as a cryptographer at Bletchley Park breaking Italian and Japanese codes. At the end of the war, Hall returned to King's College, and was soon promoted. In 1953 he was appointed to the Sadleirian Chair. His work during the 1950s proved to be extremely influential to the rapid development of group theory during the 1960s.

Hall loved poetry and recited it beautifully in Italian and Japanese, as well as English. He was interested in art, music, and botany. He was quite shy and disliked large groups of people. Hall had an incredibly broad and varied knowledge, and was respected for his integrity, intellectual standards, and judgement. He was beloved by his students.

To prove the *if* part of the theorem, the more difficult part, we need to show that if $|N(A)| \geq |A|$ for all $A \subseteq V_1$, then there is a complete matching M from V_1 to V_2. We will use strong induction on $|V_1|$ to prove this.

Basis step: If $|V_1| = 1$, then V_1 contains a single vertex v_0. Because $|N(\{v_0\})| \geq |\{v_0\}| = 1$, there is at least one edge connecting v_0 and a vertex $w_0 \in V_2$. Any such edge forms a complete matching from V_1 to V_2.

Inductive step: We first state the inductive hypothesis.

Inductive hypothesis: Let k be a positive integer. If $G = (V, E)$ is a bipartite graph with bipartition (V_1, V_2), and $|V_1| = j \leq k$, then there is a complete matching M from V_1 to V_2 whenever the condition that $|N(A)| \geq |A|$ for all $A \subseteq V_1$ is met.

Now suppose that $H = (W, F)$ is a bipartite graph with bipartition (W_1, W_2) and $|W_1| = k + 1$. We will prove that the inductive holds using a proof by cases, using two case. Case *(i)* applies when for all integers j with $1 \leq j \leq k$, the vertices in every set of j elements from W_1 are adjacent to at least $j + 1$ elements of W_2. Case *(ii)* applies when for some j with $1 \leq j \leq k$ there is a subset W_1' of j vertices such that there are exactly j neighbors of these vertices in W_2. Because either Case *(i)* or Case *(ii)* holds, we need only consider these cases to complete the inductive step.

Case (i): Suppose that for all integers j with $1 \leq j \leq k$, the vertices in every subset of j elements from W_1 are adjacent to at least $j + 1$ elements of W_2. Then, we select a vertex $v \in W_1$ and an element $w \in N(\{v\})$, which must exist by our assumption that $|N(\{v\}| \geq |\{v\}| = 1$. We delete v and w and all edges incident to them from H. This produces a bipartite graph H' with bipartition $(W_1 - \{v\}, W_2 - \{w\})$. Because $|W_1 - \{v\}| = k$, the inductive hypothesis tells us there is a complete matching from $W_1 - \{v\}$ to $W_2 - \{w\}$. Adding the edge from v to w to this complete matching produces a complete matching from W_1 to W_2.

Case (ii): Suppose that for some j with $1 \leq j \leq k$, there is a subset W_1' of j vertices such that there are exactly j neighbors of these vertices in W_2. Let W_2' be the set of these neighbors. Then, by the inductive hypothesis there is a complete matching from W_1' to W_2'. Remove these $2j$ vertices from W_1 and W_2 and all incident edges to produce a bipartite graph K with bipartition $(W_1 - W_1', W_2 - W_2')$.

We will show that the graph K satisfies the condition $|N(A)| \geq |A|$ for all subsets A of $W_1 - W_1'$. If not, there would be a subset of t vertices of $W_1 - W_1'$ where $1 \leq t \leq k + 1 - j$ such that the vertices in this subset have fewer than t vertices of $W_2 - W_2'$ as neighbors. Then, the set of $j + t$ vertices of W_1 consisting of these t vertices together with the j vertices we removed from W_1 has fewer than $j + t$ neighbors in W_2, contradicting the hypothesis that $|N(A)| \geq |A|$ for all $A \subseteq W_1$.

Hence, by the inductive hypothesis, the graph K has a complete matching. Combining this complete matching with the complete matching from W_1' to W_2', we obtain a complete matching from W_1 to W_2.

We have shown that in both cases there is a complete matching from W_1 to W_2. This completes the inductive step and completes the proof. ◁

We have used strong induction to prove Hall's marriage theorem. Although our proof is elegant, it does have some drawbacks. In particular, we cannot construct an algorithm based on this proof that finds a complete matching in a bipartite graph. For a constructive proof that can be used as the basis of an algorithm, see [Gi85].

10.2.6 Some Applications of Special Types of Graphs

We conclude this section by introducing some additional graph models that involve the special types of graph we have discussed in this section.

EXAMPLE 16 **Local Area Networks** The various computers in a building, such as minicomputers and personal computers, as well as peripheral devices such as printers and plotters, can be connected using a *local area network*. Some of these networks are based on a *star topology*, where all devices are connected to a central control device. A local area network can be represented using a complete bipartite graph $K_{1,n}$, as shown in Figure 11(a). Messages are sent from device to device through the central control device.

Links

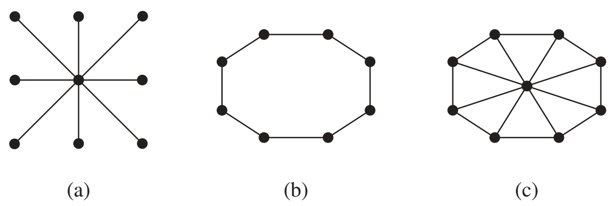


FIGURE 11 **Star, ring, and hybrid topologies for local area networks.**

Other local area networks are based on a *ring topology*, where each device is connected to exactly two others. Local area networks with a ring topology are modeled using n-cycles, C_n, as shown in Figure 11(b). Messages are sent from device to device around the cycle until the intended recipient of a message is reached.

Finally, some local area networks use a hybrid of these two topologies. Messages may be sent around the ring, or through a central device. This redundancy makes the network more reliable. Local area networks with this redundancy can be modeled using wheels W_n, as shown in Figure 11(c). ◀

EXAMPLE 17 **Interconnection Networks for Parallel Computation** For many years, computers executed programs one operation at a time. Consequently, the algorithms written to solve problems were designed to perform one step at a time; such algorithms are called **serial**. (Almost all algorithms described in this book are serial.) However, many computationally intense problems, such as weather simulations, medical imaging, and cryptanalysis, cannot be solved in a reasonable amount of time using serial operations, even on a supercomputer. Furthermore, there is a physical limit to how fast a computer can carry out basic operations, so there will always be problems that cannot be solved in a reasonable length of time using serial operations.

Parallel processing, which uses computers made up of many separate processors, each with its own memory, helps overcome the limitations of computers with a single processor. **Parallel algorithms**, which break a problem into a number of subproblems that can be solved concurrently, can then be devised to rapidly solve problems using a computer with multiple processors. In a parallel algorithm, a single instruction stream controls the execution of the algorithm, sending subproblems to different processors, and directs the input and output of these subproblems to the appropriate processors.

When parallel processing is used, one processor may need output generated by another processor. Consequently, these processors need to be interconnected. We can use the appropriate type of graph to represent the interconnection network of the processors in a computer with multiple processors. In the following discussion, we will describe the most commonly used types of interconnection networks for parallel processors. The type of interconnection network used to implement a particular parallel algorithm depends on the requirements for exchange of data between processors, the desired speed, and, of course, the available hardware.

The simplest, but most expensive, network-interconnecting processors include a two-way link between each pair of processors. This network can be represented by K_n, the complete graph on n vertices, when there are n processors. However, there are serious problems with this type of interconnection network because the required number of connections is so large. In reality, the number of direct connections to a processor is limited, so when there are a large number of

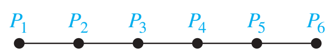


FIGURE 12 **A linear array for six processors.**

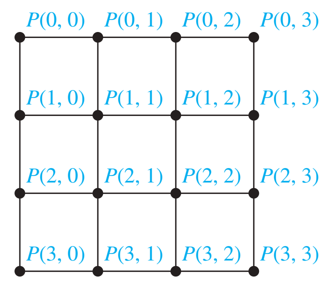


FIGURE 13 **A mesh network for 16 processors.**

processors, a processor cannot be linked directly to all others. For example, when there are 64 processors, $C(64, 2) = 2016$ connections would be required, and each processor would have to be directly connected to 63 others.

On the other hand, perhaps the simplest way to interconnect n processors is to use an arrangement known as a **linear array**. Each processor P_i, other than P_1 and P_n, is connected to its neighbors P_{i-1} and P_{i+1} via a two-way link. P_1 is connected only to P_2, and P_n is connected only to P_{n-1}. The linear array for six processors is shown in Figure 12. The advantage of a linear array is that each processor has at most two direct connections to other processors. The disadvantage is that it is sometimes necessary to use a large number of intermediate links, called **hops**, for processors to share information.

The **mesh network** (or **two-dimensional array**) is a commonly used interconnection network. In such a network, the number of processors is a perfect square, say $n = m^2$. The n processors are labeled $P(i, j)$, $0 \leq i \leq m - 1$, $0 \leq j \leq m - 1$. Two-way links connect processor $P(i, j)$ with its four neighbors, processors $P(i \pm 1, j)$ and $P(i, j \pm 1)$, as long as these are processors in the mesh. (Note that four processors, on the corners of the mesh, have only two adjacent processors, and other processors on the boundaries have only three neighbors. Sometimes a variant of a mesh network in which every processor has exactly four connections is used; see Exercise 74.) The mesh network limits the number of links for each processor. Communication between some pairs of processors requires $O(\sqrt{n}) = O(m)$ intermediate links. (See Exercise 75.) The graph representing the mesh network for 16 processors is shown in Figure 13.

One important type of interconnection network is the hypercube. For such a network, the number of processors is a power of 2, $n = 2^m$. The n processors are labeled $P_0, P_1, \ldots, P_{n-1}$. Each processor has two-way connections to m other processors. Processor P_i is linked to the processors with indices whose binary representations differ from the binary representation of i in exactly one bit. The hypercube network balances the number of direct connections for each processor and the number of intermediate connections required so that processors can communicate. Many computers have been built using a hypercube network, and many parallel algorithms have been devised that use a hypercube network. The graph Q_m, the m-cube, represents the hypercube network with $n = 2^m$ processors. Figure 14 displays the hypercube network for eight processors. (Figure 14 displays a different way to draw Q_3 than was shown in Figure 6.) ◀

10.2.7 New Graphs from Old

Sometimes we need only part of a graph to solve a problem. For instance, we may care only about the part of a large computer network that involves the computer centers in New York, Denver, Detroit, and Atlanta. Then we can ignore the other computer centers and all telephone lines not linking two of these specific four computer centers. In the graph model for the large

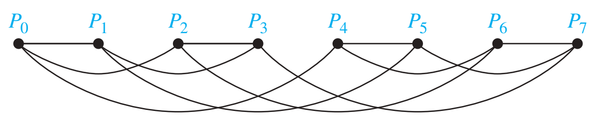


FIGURE 14 A hypercube network for eight processors.

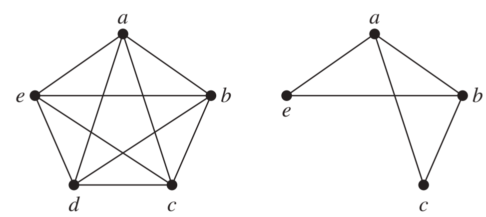


FIGURE 15 A subgraph of K_5.

network, we can remove the vertices corresponding to the computer centers other than the four of interest, and we can remove all edges incident with a vertex that was removed. When edges and vertices are removed from a graph, without removing endpoints of any remaining edges, a smaller graph is obtained. Such a graph is called a **subgraph** of the original graph.

Definition 7 A *subgraph of a graph* $G = (V, E)$ is a graph $H = (W, F)$, where $W \subseteq V$ and $F \subseteq E$. A subgraph H of G is a *proper subgraph* of G if $H \neq G$.

Given a set of vertices of a graph, we can form a subgraph of this graph with these vertices and the edges of the graph that connect them.

Definition 8 Let $G = (V, E)$ be a simple graph. The **subgraph induced** by a subset W of the vertex set V is the graph (W, F), where the edge set F contains an edge in E if and only if both endpoints of this edge are in W.

EXAMPLE 18 The graph G shown in Figure 15 is a subgraph of K_5. If we add the edge connecting c and e to G, we obtain the subgraph induced by $W = \{a, b, c, e\}$. ◀

REMOVING OR ADDING EDGES OF A GRAPH Given a graph $G = (V, E)$ and an edge $e \in E$, we can produce a subgraph of G by removing the edge e. The resulting subgraph, denoted by $G - e$, has the same vertex set V as G. Its edge set is $E - \{e\}$. Hence,

$$G - e = (V, E - \{e\}).$$

Similarly, if E' is a subset of E, we can produce a subgraph of G by removing the edges in E' from the graph. The resulting subgraph has the same vertex set V as G. Its edge set is $E - E'$.

We can also add an edge e to a graph to produce a new larger graph when this edge connects two vertices already in G. We denote by $G + e$ the new graph produced by adding a new edge e, connecting two previously nonincident vertices, to the graph G. Hence,

$$G + e = (V, E \cup \{e\}).$$

The vertex set of $G + e$ is the same as the vertex set of G and the edge set is the union of the edge set of G and the set $\{e\}$. (See Example19 for examples of removing an edge from a graph and adding an edge to a graph.)

EDGE CONTRACTIONS Sometimes when we remove an edge from a graph, we do not want to retain the endpoints of this edge as separate vertices in the resulting subgraph. In such a case we perform an **edge contraction**, which removes an edge e with endpoints u and v and merges u

and w into a new single vertex w, and for each edge with u or v as an endpoint replaces the edge with one with w as endpoint in place of u or v and with the same second endpoint. Hence, the contraction of the edge e with endpoints u and v in the graph $G = (V, E)$ produces a new graph $G' = (V', E')$ (which is not a subgraph of G), where $V' = V - \{u, v\} \cup \{w\}$ and E' contains the edges in E which do not have either u or v as endpoints and an edge connecting w to every neighbor of either u or v in V. For example, the contraction of the edge connecting the vertices e and c in the graph G_1 in Figure 16 produces a new graph G_1' with vertices a, b, d, and w. As in G_1, there is an edge in G_1' connecting a and b and an edge connecting a and d. There also is an edge in G_1' that connects b and w that replaces the edges connecting b and c and connecting b and e in G_1 and an edge in G_1' that connects d and w replacing the edge connecting d and e in G_1. (Also, see Example 19 for an example of the contraction of an edge in a graph.)

REMOVING VERTICES FROM A GRAPH When we remove a vertex v and all edges incident to it from $G = (V, E)$, we produce a subgraph, denoted by $G - v$. Observe that $G - v = (V - \{v\}, E')$, where E' is the set of edges of G not incident to v. Similarly, if V' is a subset of V, then the graph $G - V'$ is the subgraph $(V - V', E')$, where E' is the set of edges of G not incident to a vertex in V'. (See Example 19 for an example of the removal of a vertex from a graph.)

EXAMPLE 19 Figure 16 displays an undirected graph G with four different graphs that are the result of different operations on G. These are:

(a) $G - \{b, c\}$, constructed from G by removing the edge $\{b, c\}$
(b) $G + \{e, d\}$, constructed from G by adding the edge $\{e, d\}$
(c) the contraction of G, constructed from G by replacing the edge $\{b, c\}$ with a new vertex f, and replacing the edges $\{c, d\}$, $\{a, b\}$, $\{b, e\}$, and $\{c, e\}$ with the new edges $\{a, f\}$, $\{f, d\}$, and $\{f, e\}$
(d) $G - c$, constructed from G by removing the vertex c and the edges $\{b, c\}$, $\{c, d\}$ and $\{c, e\}$ ◀

GRAPH UNIONS Two or more graphs can be combined in various ways. The new graph that contains all the vertices and edges of these graphs is called the **union** of the graphs. We will give a more formal definition for the union of two simple graphs.

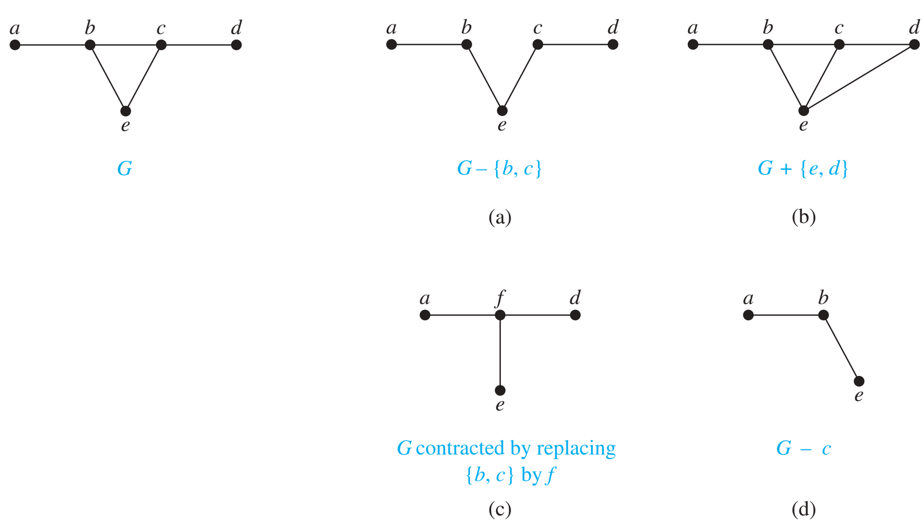


FIGURE 16 **The graph G and four graphs resulting from different operations on G.**

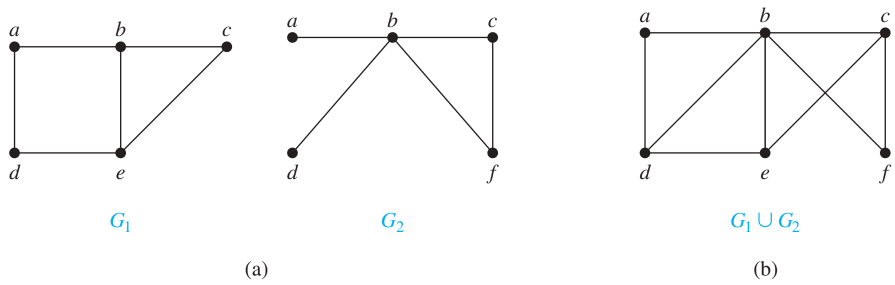


FIGURE 17 **(a) The simple graphs G_1 and G_2. (b) Their union $G_1 \cup G_2$.**

Definition 9 The *union* of two simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$. The union of G_1 and G_2 is denoted by $G_1 \cup G_2$.

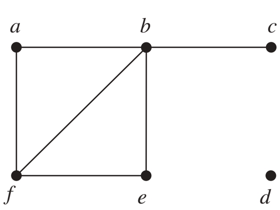
EXAMPLE 20 Find the union of the graphs G_1 and G_2 shown in Figure 17(a). ◀

Solution: The vertex set of the union $G_1 \cup G_2$ is the union of the two vertex sets, namely, $\{a, b, c, d, e, f\}$. The edge set of the union is the union of the two edge sets. The union is displayed in Figure 17(b).

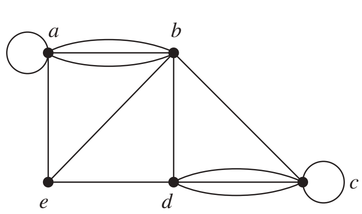
Exercises

In Exercises 1–3 find the number of vertices, the number of edges, and the degree of each vertex in the given undirected graph. Identify all isolated and pendant vertices.

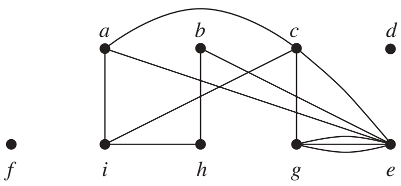
1.



2.



3.



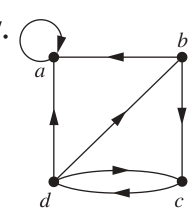
4. Find the sum of the degrees of the vertices of each graph in Exercises 1–3 and verify that it equals twice the number of edges in the graph.

5. Can a simple graph exist with 15 vertices each of degree five?

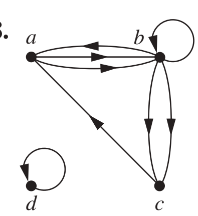
6. Show that the sum, over the set of people at a party, of the number of people a person has shaken hands with, is even. Assume that no one shakes his or her own hand.

In Exercises 7–9 determine the number of vertices and edges and find the in-degree and out-degree of each vertex for the given directed multigraph.

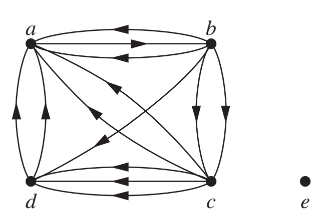
7.



8.



9.



10. For each of the graphs in Exercises 7–9 determine the sum of the in-degrees of the vertices and the sum of the out-degrees of the vertices directly. Show that they are both equal to the number of edges in the graph.