Chapter 2
Graphs

Abstract Graphs are discrete structures that consist of vertices and edges connecting some of these vertices. Graphs have many applications in Mathematics, Computer Science, Engineering, Bioinformatics, and many other disciplines. Graphs are frequently used to model a communication network where computational nodes of a network are represented by vertices and the communication links between the nodes are represented by edges of the graph. In this chapter, we will review basic concepts in graph theory in relation to the modeling of a distributed system.

2.1 Definition of Graphs

Definition 2.1 (Graph) A graph is a tuple $G(V, E)$ where V is a nonempty set of *vertices* (or *nodes*) and E is a set of *edges*. Each edge has either one or two vertices as endpoints, that is, each edge is either a one- or two-element subset of V.

The vertex set V of a graph G may be infinite, in which case the graph is called an *infinite graph*, and a graph with a finite vertex set is called a *finite* graph. In this book, we will only consider finite graphs. For the graph $G = (V, E)$ and $v \in V$, the edge $e = \{v\}$ is called a *self-loop*. An edge is identified by the two vertices, and the edge is said to be incident to the vertices. For example, edge $e = \{v_1, v_2\}$, sometimes shown as $e = v_1v_2$ or $e_{v_1v_2}$, is incident to the vertices v_1 and v_2. The number of vertices of a graph ($|V|$) is called its *order*, and the number of its edges ($|E|$) is called its *size*. We will use literals n for the order and m for the size of a graph.

A graph that contains multiple edges connecting the same vertices is called a *multigraph*. A graph that does not contain edges that are self-loops and is not a multigraph is called a *simple graph*. We will only consider simple graphs in this book.

Definition 2.2 (Vertex Adjacency) Let $G(V, E)$ be a graph. Two vertices v_1 and v_2 are said to be *adjacent* if there exists an edge $e \in E$ that connects them so that $e = \{v_1, v_2\}$.

Definition 2.3 (Edge Adjacency) For a graph $G(V, E)$, two edges e_1 and e_2 are said to be adjacent if there exists a vertex v that is incident to (connects) both edges.

K. Erciyes, *Distributed Graph Algorithms for Computer Networks*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5173-9_2,
© Springer-Verlag London 2013

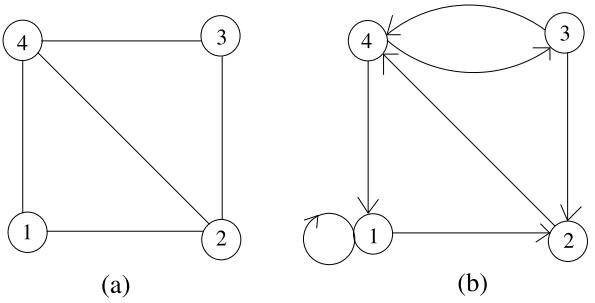


Fig. 2.1 (**a**) An undirected simple graph; (**b**) a directed multigraph

Based on these definitions, we can now define the neighborhood of a vertex as follows.

Definition 2.4 (Neighborhood) Given $G(V, E)$, the *neighborhood* of a vertex $v \in V$ is the set of vertices that are adjacent to v. Formally,

$$N(v) = \{u \in V : e(u, v) \in E\}.$$

$N(v)$ is usually called the *open neighborhood* of v, whereas $N[v] = N(v) \cup \{v\}$ is called the *closed neighborhood* of v, that is, the union of all neighbors of v and itself. The vertices of a graph are drawn as circles, and edges are the lines joining these vertices as shown in the example graph of Fig. 2.1(a), where $V = \{1, 2, 3, 4\}$ and $E = \{\{1, 2\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 1\}\}$. The neighborhood sets for vertex 2 are $N(2) = \{1, 3, 4\}$ and $N[2] = \{1, 2, 3, 4\}$. We will mostly use numbers to represent the vertices, unless this complicates description of an algorithm, in which case we will use letters.

Definition 2.5 (Degree) The degree of $v \in V$, $\deg(v)$, is the number of edges plus twice the number of self-loop edges incident to v.

The maximum degree of a graph is denoted by $\Delta(G)$, and the minimum degree by $\delta(G)$. $\Delta(G)$ of the graph in Fig. 2.1(a) is 3, and $\delta(G)$ is 2.

Up to now, we have considered *undirected graphs* that have *undirected edges*. However, in certain applications, such as the representation of data flow in computer networks, it may be required to assign directions to edges, in which case *directed graphs* are obtained.

Definition 2.6 (Directed Graph) A *directed graph* (*digraph*) $G(V, E)$ consists of a nonempty set of vertices V and a set of directed edges E where each $e \in E$ is associated with an ordered set of vertices.

An edge e that is associated with the ordered pair (u, v) is described as starting from u and ending at v. Figure 2.1(b) shows a digraph with $V = \{1, 2, 3, 4\}$ and $E = \{\{1, 1\}, \{1, 2\}, \{2, 4\}, \{3, 2\}, \{3, 4\}, \{4, 3\}, \{4, 1\}\}$.

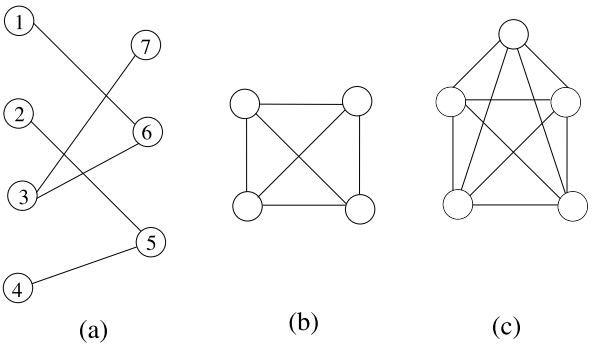


Fig. 2.2 (**a**) A bipartite graph; (**b**) K_3; (**c**) K_4

Definition 2.7 (In-Degree, Out-Degree) The *in-degree* of a vertex v in a digraph G is the total number of edges in E that end at v. The *out-degree* of v is the total number of edges in E that start from v. We will denote the in-degree of v by $\deg_{\text{in}}(v)$ and the out-degree by $\deg_{\text{out}}(v)$.

2.1.1 Special Graphs

We will describe some special graphs such as a *complete graph*, *bipartite graph*, and the *complement of a graph* in this part.

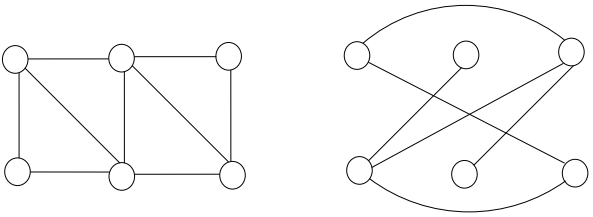
Definition 2.8 (Complete Graph) For the graph $G(V, E)$, if $\forall v \in V$, $N(v) = V \setminus \{v\}$, that is, if every vertex is connected to all other vertices of G, then G is called a *complete* graph. For a graph G with n vertices, the complete graph is denoted by K_n. For $K_n(V, E)$, $|E| = n(n-1)/2$.

Definition 2.9 (Bipartite Graphs) A graph $G(V, E)$ is called *bipartite* if V can be partitioned into two disjoint sets V_1 and V_2 such that every edge of G joins a vertex in V_1 to a vertex in V_2.

A bipartite graph with $V_1 = \{1, 2, 3, 4\}$ and $V_2 = \{5, 6, 7\}$ is shown in Fig. 2.2(a), and K_4 and K_5 are shown in Fig. 2.2(b) and (c).

Definition 2.10 (Complement of a Graph) The complement of a graph $G(V, E)$ is the graph $H(V, E')$ such that $e = \{v_1, v_2\} \in E'$ if and only if $e = \{v_1, v_2\} \notin E$. The complement of G is denoted G' or $\bar{G}$.

A graph G and its complement are shown in Fig. 2.3. A weighted graph $G(V, E, w)$ is a graph that has weights associated with edges, that is, $w : E \to \mathbb{R}$. Weighted graphs are frequently used to model communication networks as associated weights for edges may represent communication costs of sending messages over the links represented by the edges.



(a) (b)

Fig. 2.3 (**a**) A graph $G(V, E)$. (**b**) Its complement $G'(V, E')$

2.1.2 Graph Representations

In order to be able to perform some computation on graphs, they have to be represented in a format suitable for processing in a computer. Two important methods of representation are the *adjacency matrices* and *adjacency lists*.

Definition 2.11 (Adjacency Matrix) The *adjacency matrix* of a graph $G(V, E)$ with n vertices is an $n \times n$ matrix which has entry 1 at element (i, j) if there is an edge connecting vertex i to vertex j and 0 otherwise.

Definition 2.12 (Incidence Matrix) The *incidence matrix* of a graph $G(V, E)$ with n vertices and m edges is an $n \times m$ matrix which has entry 1 at element (i, j) if vertex i is incident to edge j and 0 otherwise.

Definition 2.13 (Adjacency List) The *adjacency list* of a graph $G(V, E)$ with n vertices is a list of n elements where each element consists of a vertex $v \in V$ and its neighbors connected using linked lists.

Figure 2.4 displays the adjacency matrix and the adjacency list of a graph.

2.2 Walks, Paths and Cycles

Definition 2.14 (Walk) A $walk$ $w = (v_1, e_1, v_2, e_2, \ldots, v_n, e_n, v_n + 1)$ in G is an alternating sequence of vertices and edges in V and E, respectively, such that for all $i = 1, \ldots, n$, $\{v_i, v_{i+1}\} = e_i$. A walk is called *closed* if $v_1 = v_{n+1}$ and *open* otherwise.

Definition 2.15 (Trail, Tour) A $trail$ in G is a walk in G where no edge is repeated and, a *tour* is a closed trail. An *Eulerian trail* is a trail that contains exactly one copy of each edge in E, and an *Eulerian tour* is a closed trail (tour) that contains exactly one copy of each edge

Definition 2.16 (Path) A *path* p from a vertex u to vertex v in graph G is a sequence of edges $e_1, \ldots, e_n$ such that each consecutive edge is incident to consecutive vertices along the path. The length of p is the number of edges it contains. When G

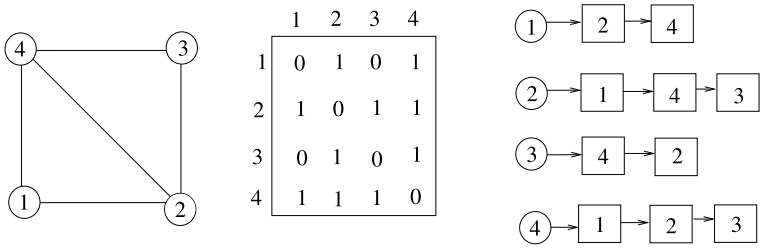


Fig. 2.4 (**a**) A graph $G(V, E)$. (**b**) Its adjacency matrix representation. (**c**) Its adjacency list representation

is simple, a path can be represented by the set of vertices $v_1, \ldots, v_n$ that it passes through (traverses). The path is called a *circuit* if it starts and ends at the same vertex. A *Hamiltonian Path* is a path that contains each vertex in V once. Alternatively, we can say that a path is a nontrivial walk with no edges and vertices repeated.

Definition 2.17 (Cycle) A *cycle* is a circuit of length of at least 3 and with no repeated edges except the first and last vertices. A *Hamiltonian cycle* is a cycle in a graph containing every vertex.

Definition 2.18 (Hamiltonian/Eulerian Graph) A graph $G = (V, E)$ is said to be *Hamiltonian* if it contains a Hamiltonian cycle and *Eulerian* if it contains an Eulerian tour.

A connected graph G is Eulerian if and only if every vertex of G has even degree. A connected graph G has Euler Trail if and only if the number of vertices with odd degree is less than or equal to 2. Figure 2.5 shows Hamiltonian Path, Hamiltonian Cycle, Eulerian Trail, and Eulerian Cycle. In (c), there are two odd-degree vertices as 2 and 8, and therefore an Eulerian Trail exists as shown. In (d), all vertices have even degrees, so an Eulerian Cycle exists as illustrated.

2.2.1 Diameter, Radius, Circumference, and Girth

Definition 2.19 (Distance) For a graph $G(V, E)$, the distance between the two vertices v_1 and v_2 in V is the length of the shortest walk beginning at v_1 and ending at v_2, provided that such a walk exists. We will write $d_G(v_1, v_2)$ to denote the distance between v_1 and v_2 in G.

Definition 2.20 (Diameter, Eccentricity, Radius) The *diameter* of G (diam(G)) is the length of the greatest distance in G. The *eccentricity* of v_1 is the maximum

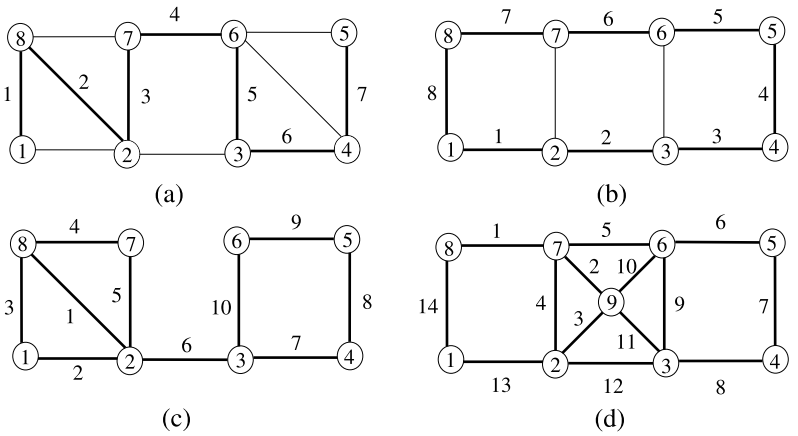


Fig. 2.5 (**a**) A Hamiltonian trail through vertices 1, 8, 2, 7, 6, 3, 4, 5. (**b**) A Hamiltonian path through vertices 1, 2, 3, 4, 5, 6, 7, 8, 1. (**c**) An Eulerian trail through vertices 8, 2, 1, 8, 7, 2, 3, 4, 5, 6, 3. (**d**) An Eulerian tour through vertices 8, 7, 9, 2, 7, 6, 5, 4, 3, 6, 9, 3, 2, 1, 8, all shown by *bold lines* and each edge labeled in sequence

distance from v_1 to any other vertex v_2 in V. The *radius* of G is the minimum eccentricity of vertices of G.

Definition 2.21 (Girth) For a graph $G(V, E)$, the *girth* of G is the length of the shortest cycle, provided that there is a cycle. When G does not have any cycle, the girth is defined as 0.

Definition 2.22 (Circumference) For a graph $G(V, E)$, the *circumference* of G is the length of the longest cycle, provided that there is a cycle in G. When G does not have any cycle, the circumference is defined as ∞.

The diameter of the graph in Fig. 2.5(a) is 4, for example, as the distance between vertices 1 and 5 through vertices 2–7–6. We will see that diam(G) is an important parameter in the determination of time complexities of distributed algorithms as it provides an upper bound on the time that a message is communicated between the two farthest points of a network graph.

2.3 Subgraphs

Certain applications may require finding solutions to a problem by computing the solution for small parts of the graph iteratively and then combining these partial solutions to obtain the final solutions. Informally, a smaller part of the graph is called a *subgraph*.

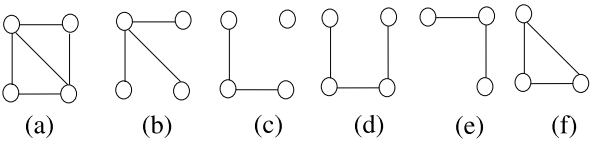


Fig. 2.6 (**a**) A graph G. (**b**)–(**d**) Spanning subgraphs of G. (**e**), (**f**) Subgraphs of G

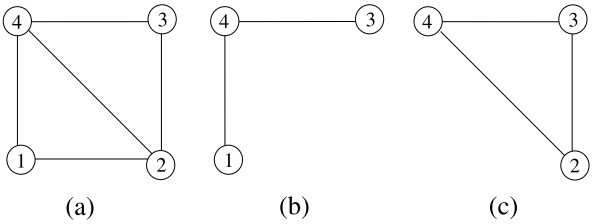


Fig. 2.7 (**a**) A graph G. (**b**) Edge-induced graph of G of edges $\{1, 4\}$, $\{4, 3\}$. (**c**) Vertex-induced graph of G of vertices 2, 3, 4

Definition 2.23 (Subgraph, Spanning Subgraph) A graph $H = (V', E')$ is called a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$, with u and $v \in V'$; $\forall\{u, v\} \in E'$, that is, all vertices of H are also vertices of G and all edges of H are also edges of G. If $V' = V$, which means that H includes (covers) all vertices of G, then H is called a *spanning subgraph* of G.

Figure 2.6 shows the subgraphs of a graph.

Definition 2.24 (Edge-Induced Subgraph, Vertex-Induced Subgraph) Given an edge set $E' \subseteq E$, the edge induced subgraph by E' is $H = (V', E')$ where $v \in V'$ if and only if it is incident to an edge in E'. Similarly, given a vertex set $V' \subseteq V$, the vertex induced subgraph by V' is $H = (V', E')$ where $\{v_1, v_2\} \in E'$ if and only if both v_1 and v_2 are in V'.

Figure 2.7 shows the edge-induced and vertex-induced subgraphs of a graph.

2.4 Connectivity

An important property of a communication network is its capacity to withstand node and link failures. For example, it may be required to know the largest number of link failures that result in a disconnected network where there is no walk between every pair of computing nodes. Similarly, in graphs, we may need to determine the number of edge removals that will result in a disconnected network. Connectivity of a network is the determination of such parameters. Also, vertex and edge deletion methods are important in some of the algorithms that require removing a vertex from the graph at each iteration; we will see some of them in Part II.

Definition 2.25 (Connectedness) A graph $G(V, E)$ is *connected* if there is a walk between any pair of vertices v_1 and v_2. A digraph G is *strongly connected* if for every walk from every vertex $v_1 \in V$ to any vertex $v_2 \in V$, there is also a walk from v_2 to v_1 [3].

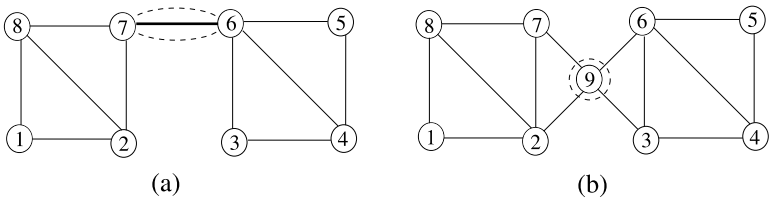


Fig. 2.8 (**a**) A bridge. (**b**) A cutpoint. Both are shown by *dashed lines*

Definition 2.26 (Component) A component of a graph $G(V, E)$ is a subgraph G' of G where any pair of vertices in G' is connected. A connected graph G has only one component which is itself.

Definition 2.27 (Edge Deletion Graph) For a graph $G(V, E)$ and $E' \subset E$, the graph G' formed after deleting the edges in E' from G is the subgraph induced by the edge set $E \setminus E'$, which is denoted $G' = G - E'$.

Definition 2.28 (Vertex Deletion Graph) For the graph $G(V, E)$ and $V' \subset V$, the graph G' formed after deleting the vertices in V' from G is the subgraph induced by the vertex set $V \setminus V'$, which is denoted $G' = G - V'$.

2.4.1 Cutpoints and Bridges

Definition 2.29 (Cutpoint) For a graph $G(V, E)$, a vertex $v \in V$ is a *cutpoint* of G if $G - v$ has more components than G has. If G is connected, $G - v$ is disconnected.

Definition 2.30 (Bridge, Cutset) For a graph $G(V, E)$, a *bridge* is an edge $e \in E$ deletion of which increases the number of components of G. A minimal set of edges whose deletion disconnects G is called a *cutset* in G.

The deletion of a bridge from a connected graph G provides two disconnected components of G.

Definition 2.31 (Block) A *block* of a graph G is its maximal subgraph that is connected and contains no cutpoints.

Figure 2.8 displays a bridge and a cutpoint of a graph. The subgraphs defined by vertices 1, 2, 7, 8 and 3, 4, 5, 6 are also blocks.

Definition 2.32 (Connectivity) The *vertex connectivity* (or just the *connectivity*) $\mathcal{K}$ of a graph G is the minimum number of vertices whose removal from G results in either a disconnected graph or a single vertex. The *edge connectivity* $\mathcal{E}(G)$ is defined as the minimum number of edges whose removal disconnects G.

2.5 Trees

Trees are important data structures in Computer Science as they have many applications such as database implementation, hereditary trees in bioinformatics, etc. A tree of a graph G also provides a graph with less edges and therefore with less communication links of the network. We will see many example algorithms to construct trees and implement distributed algorithms over the trees.

Definition 2.33 (Forest, Tree) A graph that contains no cycles is called *acyclic*. If $G = (V, E)$ is an acyclic graph and has more than one component, G is called a *forest*. If G has one component, then G is called a *tree*. Directed trees and forests are acyclic directed graphs.

The following are equivalent to describe a tree T:

- T is a tree;
- T contains no cycles and has $n - 1$ edges;
- T is connected and has $n - 1$ edges;
- T is connected, and each edge is a bridge;
- Any two vertices of T are connected by exactly one path;
- T contains no cycles, but the addition of any new edge creates exactly one cycle.

Definition 2.34 (Rooted Tree, parent, child, leaf) A tree is *rooted* if it has a designated vertex, called the *root*, in which case the edges have a natural orientation, toward or away from the root. In a rooted tree, the *parent* of a vertex is the vertex connected to it on the path to the root; every vertex except the root has a unique parent. A *child* of a vertex v is a vertex of which v is the parent. A *leaf* is a vertex without children.

Definition 2.35 (Spanning Forest, Spanning Tree) For graph $G(V, E)$, if $H(V', E')$ is an acyclic subgraph of G where $V' = V$, then H is called a *spanning forest* of G. If H has one component, it is called a *spanning tree* of G.

2.5.1 Minimum Spanning Trees

Definition 2.36 (Minimum Spanning Tree) For a weighted graph $G(V, E)$ where weights are associated with edges, a spanning tree H of G is called a *minimum spanning tree* of G if the total sum of the weights of its edges is minimal among all possible spanning trees of G.

If all weights of the edges of a graph G are distinct, then there is exactly one spanning tree of G. Figure 2.9 displays a possible spanning tree of a graph and its rooted minimum spanning tree.

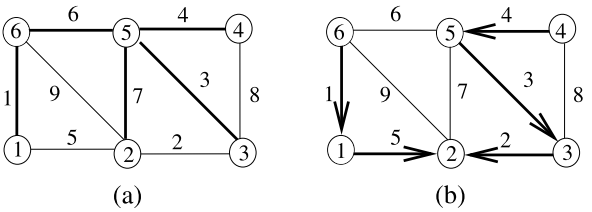


Fig. 2.9 (**a**) A spanning tree. (**b**) The minimum spanning tree rooted at vertex 2

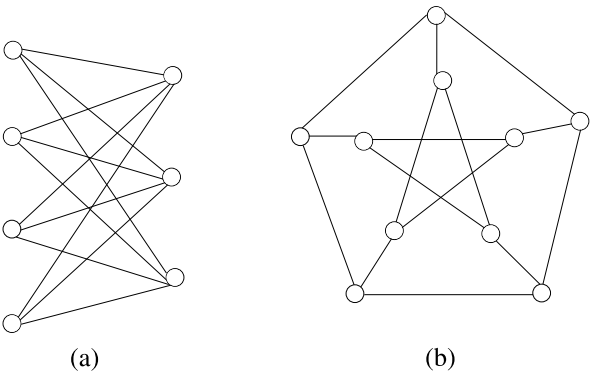


Fig. 2.10 (**a**) Complete bipartite graph $K_{4,3}$. (**b**) Petersen graph

2.6 Chapter Notes

The classical book on graph theory is by Harary [4] dating back to 1979. The textbook by West [5] is a more updated presentation of the topic. A thorough treatment of the topic is provided in the recent book by Bondy and Murty [1], and an informal presentation is provided in [2].

We have reviewed some of the basic concepts in graph theory. We will use undirected graphs to model the computer networks and develop distributed algorithms using this model. In Part III, we will need to modify this model to include ad hoc wireless networks.

2.6.1 Exercises

1. Show that the sum of the degrees of the vertices of an undirected graph is even. Show also that the number of odd degree vertices of an undirected graph is even.
2. For a bipartite graph $G(P, Q)$ where P and Q are disjoint vertex sets, show that $\sum_{u \in P} \deg(u) = \sum_{v \in Q} \deg(v)$.
3. A *degree sequence* of a graph G is the sequence of the degrees of the vertices of G in decreasing order. Find the degree sequences of the graphs in Fig. 2.8.
4. Show that for any graph G, $\text{rad}(G) \leq \text{diam}(G) \leq 2\,\text{rad}(G)$.

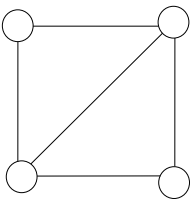


Fig. 2.11 An example graph for Exercises 9 and 12

5. A simple graph G is called *regular* if all vertices of G have the same degree. In an *n-regular* graph G, all vertices have a degree of n. Determine the values of n for K_n and $K_{m,n}$ for these graphs to be n-regular.
6. Let G be a graph that has n vertices and m edges. Find the number of induced subgraphs and edge-induced subgraphs of G.
7. For which values of m and n does the complete bipartite graph $K_{m,n}$ have an Eulerian circuit and an Eulerian path?
8. Find the radius, girth, and diameter of the complete bipartite graph $K_{m,n}$ in terms of m and n and the Petersen graph shown in Fig. 2.10.
9. Draw all the subgraphs of the graph in Fig. 2.11.
10. Show that every tree with maximum degree k has at least k leaves.
11. A tree T with n vertices has a vertex of degree k. Prove that the longest path in T has at most $n - k + 1$ edges.
12. Find the spanning trees of the graph of Fig. 2.11.

References

1. Bondy JA, Murty USR (2008) Graph theory. Springer graduate texts in mathematics. Springer, Berlin. ISBN 978-1-84628-970-5
2. Fournier JC (2009) Graph theory and applications. Wiley, New York. ISBN 978-1-848321-070-7
3. Griffin C (2011) Graph theory. Penn State Math 485, Lecture notes. Homepage: http://www.personal.psu.edu/cxg286/Math485.pdf
4. Harary G (1979) Graph theory. Addison-Wesley, Reading
5. West DB (2001) Introduction to graph theory, 2nd edn. Prentice Hall, New York. ISBN 0-13-014400-2

http://www.springer.com/978-1-4471-5172-2

Distributed Graph Algorithms for Computer Networks

Erciyes, K.

2013, XVIII, 324 p., Hardcover
ISBN: 978-1-4471-5172-2