8.4 A Shortest-Path Algorithm

Recall (see Section 8.1) that a weighted graph is a graph in which values are assigned to the edges and that the length of a path in a weighted graph is the sum of the weights of the edges in the path. We let $w(i, j)$ denote the weight of edge (i, j). In weighted graphs, we often want to find the **shortest path** (i.e., a path having minimum length) between two given vertices. Algorithm 8.4.1, due to E. W. Dijkstra, which efficiently solves this problem, is the topic of this section.

Algorithm 8.4.1 Dijkstra's Shortest-Path Algorithm

This algorithm finds the length of a shortest path from vertex *a* to vertex *z* in a connected, weighted graph. The weight of edge (i, j) is $w(i, j) > 0$ and the label of vertex x is $L(x)$. At termination, $L(z)$ is the length of a shortest path from *a* to *z*.

Input: A connected, weighted graph in which all weights are positive; vertices *a* and *z*

Output: $L(z)$, the length of a shortest path from *a* to *z*

dijkstra(w, a, z, L) {
$L(a) = 0$
for all vertices $x \neq a$
$L(x)=\infty$
T = set of all vertices
// T is the set of vertices whose shortest distance from a has
// not been found
while $(z \in T)$ {
choose $v \in T$ with minimum $L(v)$
$T = T - \{v\}$
for each $x \in T$ adjacent to v
$L(x) = \min\{L(x), L(v) + w(v, x)\}$
}

14. }

Example 8.4.2

We show how Algorithm 8.4.1 finds a shortest path from *a* to *z* in the graph of Figure 8.4.1.

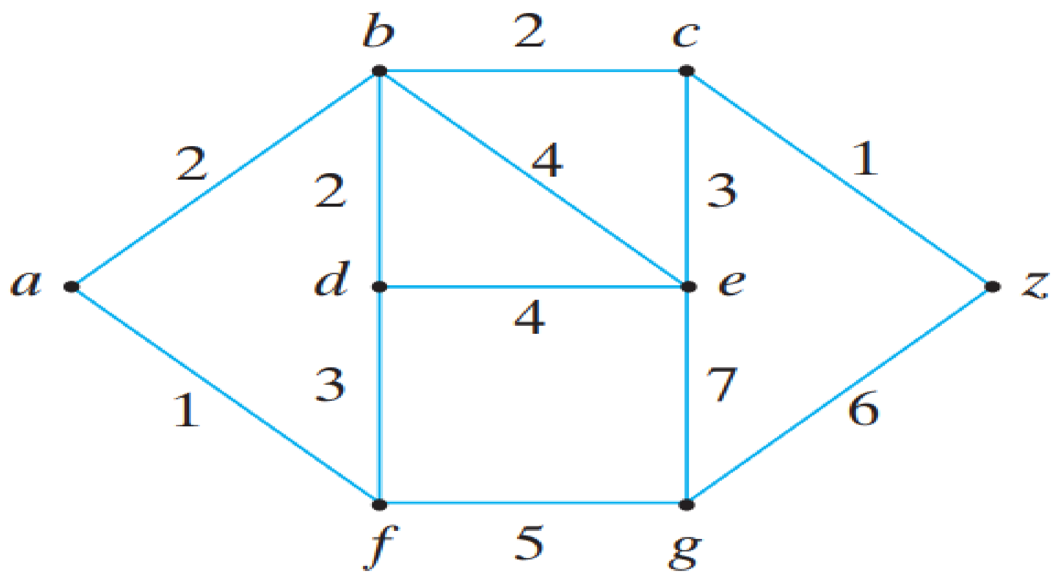


Figure 8.4.1 The graph for Example 8.4.2.

(The vertices in *T* are uncircled and have temporary labels. The circled vertices have permanent labels.) Figure 8.4.2 shows the result of executing lines 2–5.

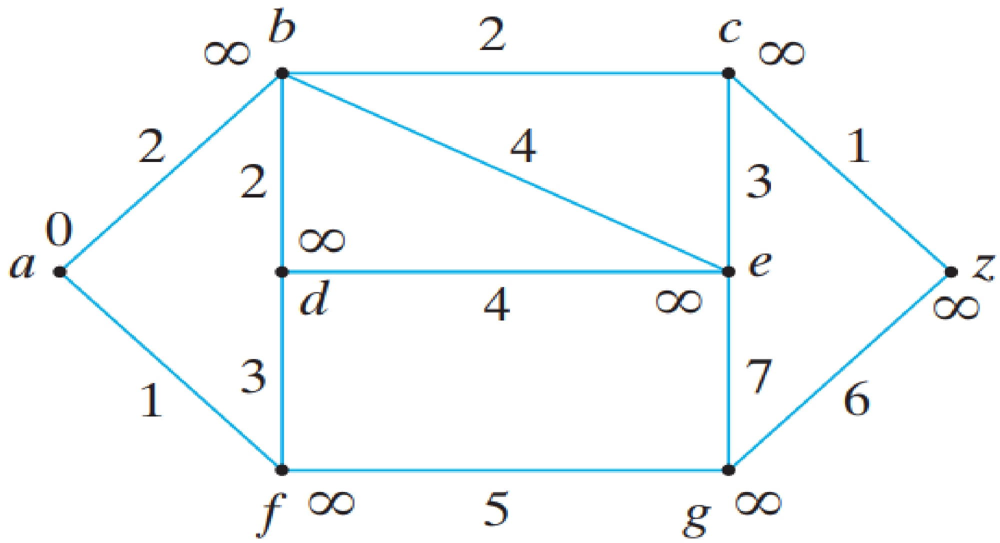


Figure 8.4.2 Initialization in Dijkstra's shortest-path algorithm.

At line 8, z is not circled. We proceed to line 9, where we select vertex a, the uncircled vertex with the smallest label, and circle it (see Figure 8.4.3).

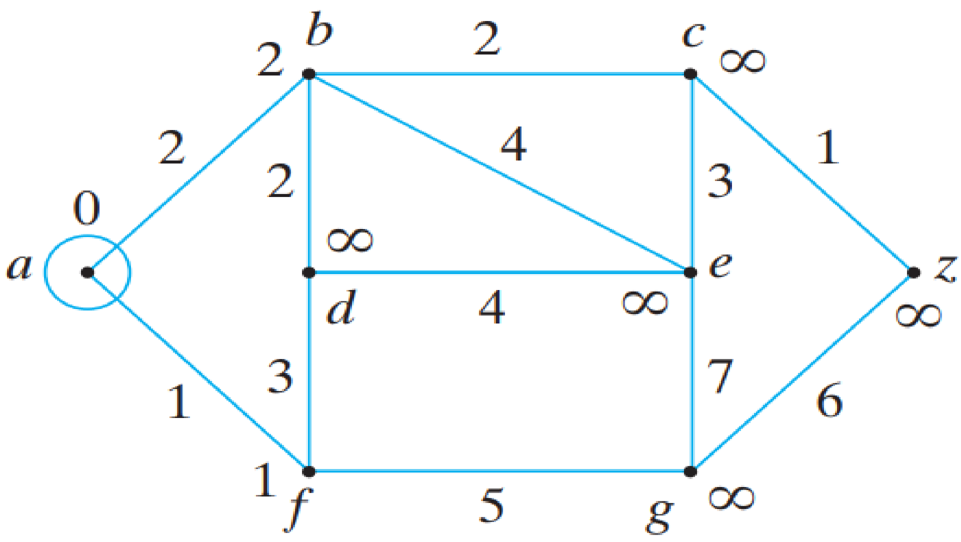


Figure 8.4.3 The first iteration of Dijkstra's shortest-path algorithm.

At lines 11 and 12 we update each of the uncircled vertices, *b* and *f*, adjacent to *a*. We obtain the new labels

$$L(b) = \min\{\infty, 0 + 2\} = 2,$$

$$L(f) = \min\{\infty, 0 + 1\} = 1$$

(see Figure 8.4.3). At this point, we return to line 8. Since *z* is not circled, we proceed to line 9, where we select vertex

f , the uncircled vertex with the smallest label, and circle it (see Figure 8.4.4).

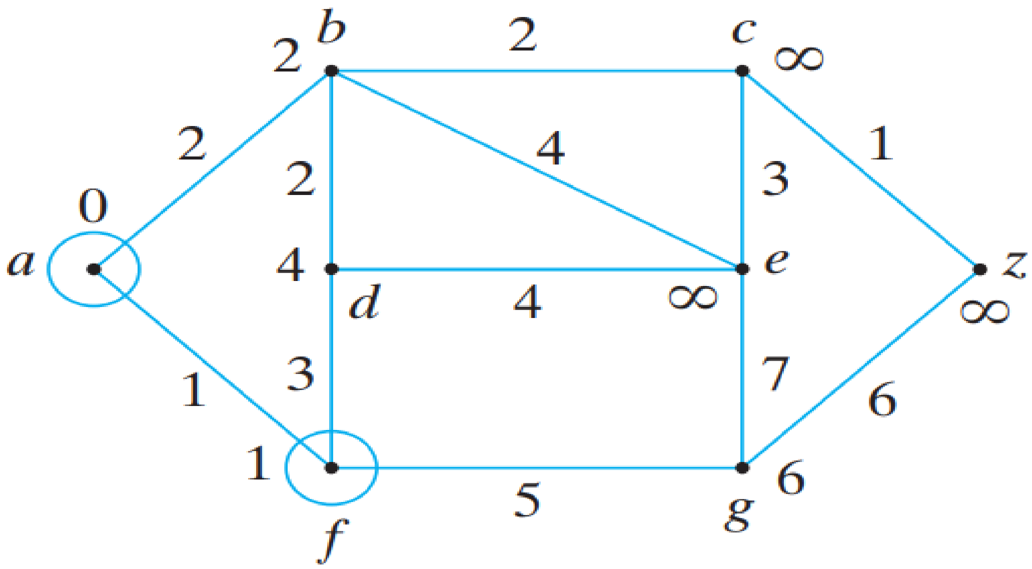


Figure 8.4.4 The second iteration of Dijkstra's shortest-path algorithm.

At lines 11 and 12 we update each label of the uncircled vertices, d and g, adjacent to f . We obtain the labels shown in Figure 8.4.4.

You should verify that the next iteration

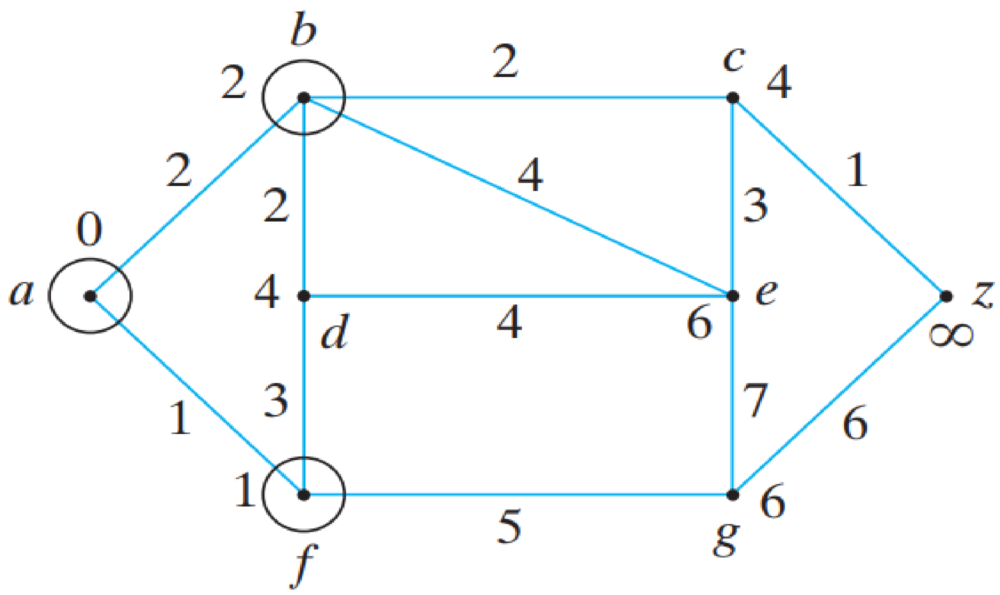


Figure 8.4.5 The third iteration of Dijkstra's shortest-path algorithm.

of the algorithm produces the labeling shown in Figure 8.4.5 and that at the termination of the algorithm, *z* is labeled 5, indicating that the length of a shortest path from *a* to *z* is 5. A shortest path is given by (*a*, *b*, *c*, *z*).

We next show that Algorithm 8.4.1 is correct. The proof hinges on the fact

that Dijkstra's algorithm finds the lengths of shortest paths from *a* in nondecreasing order.

Theorem 8.4.3 Dijkstra's shortest-path algorithm (Algorithm 8.4.1) correctly finds the length of a shortest path from *a* to *z*.

Proof

We use mathematical induction on *i* to prove that the i^{th} time we arrive at line 9, *L(v)* is the length of a shortest path from *a* to *v*. When this is proved, correctness of the algorithm follows since when *z* is chosen at line 9, *L(z)* will give the length of a shortest path from *a* to *z*.

Example 8.4.4 Find the shortest path from *a* to *z* and its length for the graph of Figure 8.4.7.

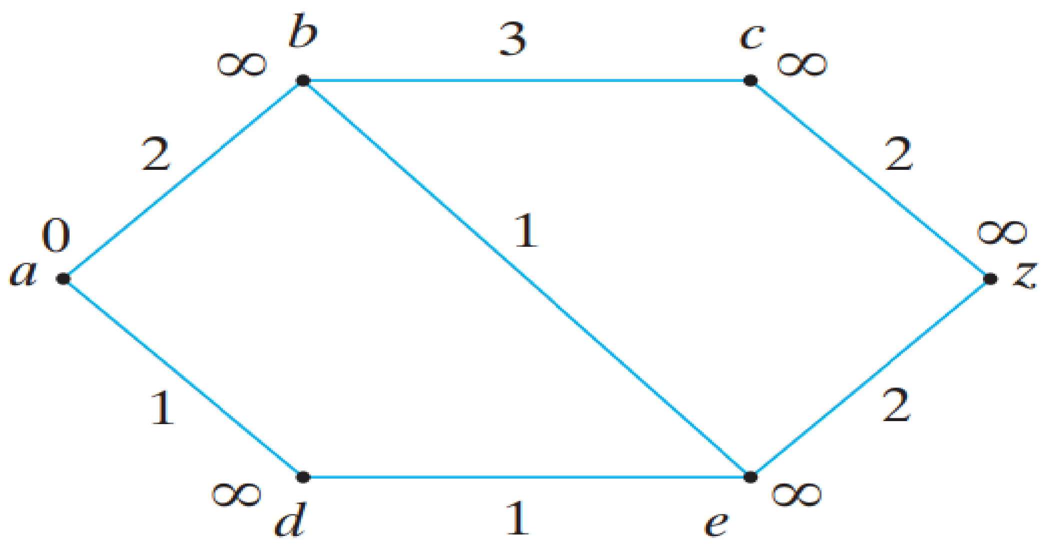


Figure 8.4.7 Initialization in Dijkstra's shortest-path algorithm.

SOLUTION: We will apply Algorithm 8.4.1 with a slight modification. In addition to circling a vertex, we will also label it with the name of the vertex from which it was labeled.

Figure 8.4.7 shows the result of executing lines 2–4 of Algorithm 8.4.1. First, we circle *a* (see Figure 8.4.8).

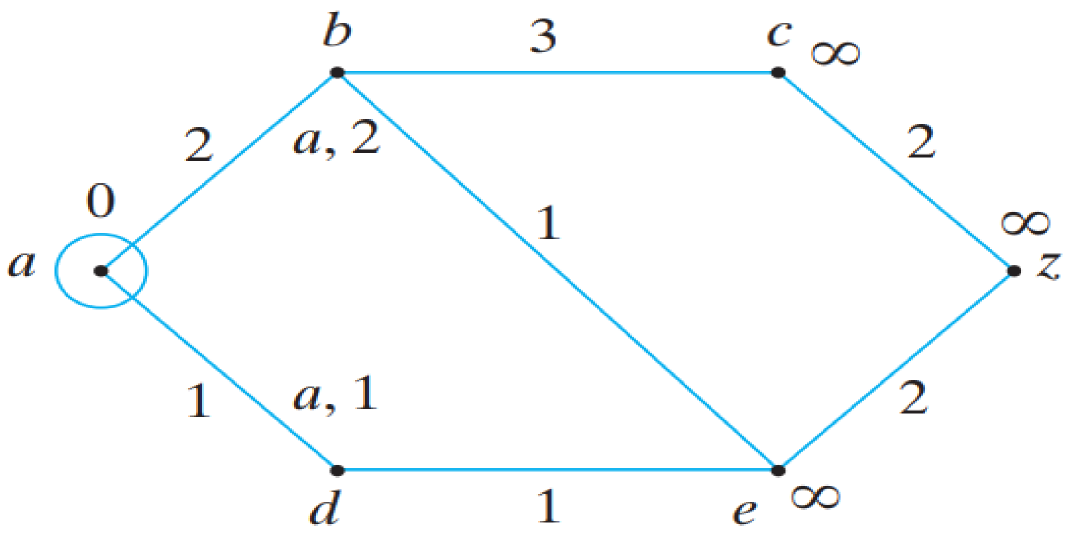


Figure 8.4.8 The first iteration of Dijkstra's shortest-path algorithm.

Next, we label the vertices *b* and *d* adjacent to *a*. Vertex *b* is labeled "*a*, 2" to indicate its value and the fact that it was labeled from *a*. Similarly, vertex *d* is labeled “*a*, 1.” Next, we circle vertex *d* and update the label of the vertex *e* adjacent to *d* (see Figure 8.4.9).

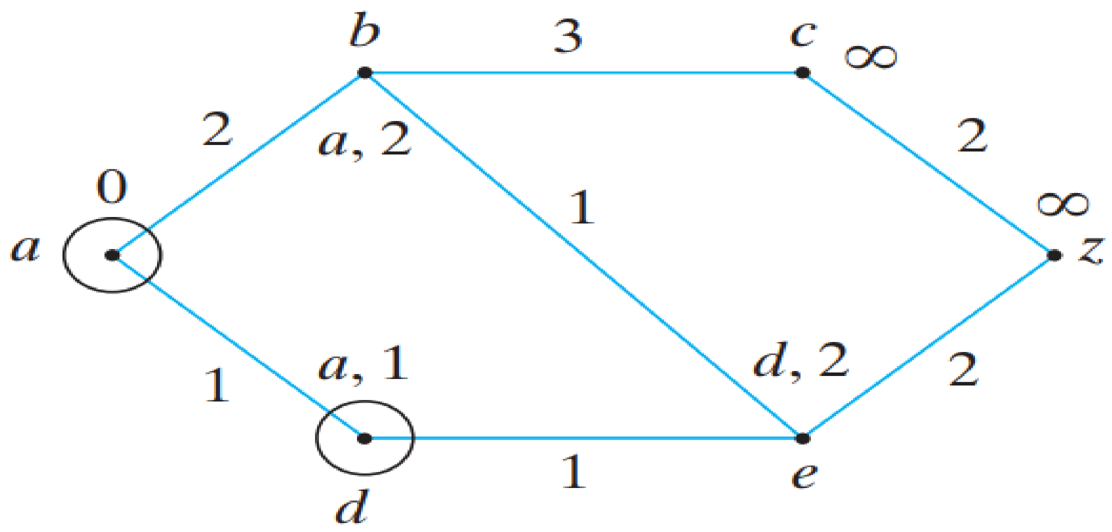


Figure 8.4.9 The second iteration of Dijkstra’s shortest-path algorithm.

Then we circle vertex *b* and update the labels of vertices *c* and *e* (see Figure 8.4.10).

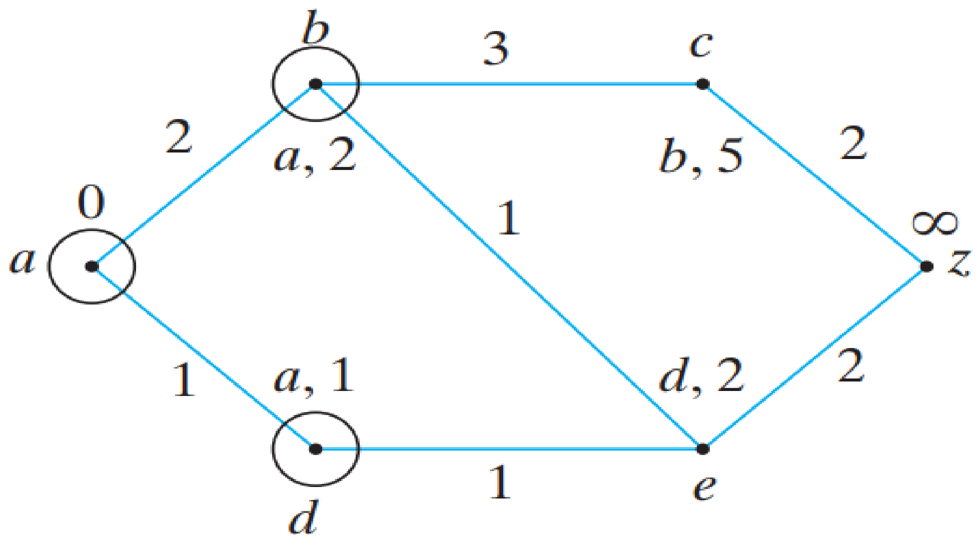


Figure 8.4.10 The third iteration of Dijkstra's shortest-path algorithm.

Next, we circle vertex *e* and update the label of vertex *z* (see Figure 8.4.11).

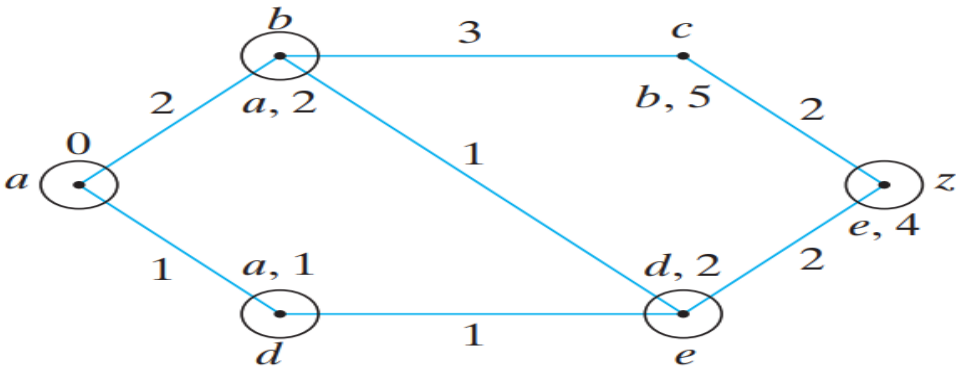


Figure 8.4.11 The conclusion of Dijkstra's shortest-path algorithm.

At this point, we may circle *z*, so the algorithm terminates. The length of the shortest path from *a* to *z* is 4. Starting at *z*, we can retrace the labels to find the shortest path *(a, d, e, z)*.

■

Our next theorem shows that Dijkstra's algorithm is $\Theta(n^2)$ in the worst case.

Theorem 8.4.5

For input consisting of an *n*-vertex, simple, connected, weighted graph, Dijkstra's algorithm (Algorithm 8.4.1) has worst-case run time $\Theta(n^2)$.

Proof:

We consider the time spent in the loops, which provides an upper bound on the total time. Line 4 is executed $O(n)$ times. Within the while loop, line 9 takes time $O(n)$ [we could find the minimum $L(v)$ by examining all the vertices in T]. The body of the for loop (line 12) takes time $O(n)$. Since lines 9 and 12 are nested within a while loop, which takes time $O(n)$, the total time for lines 9 and 12 is $O(n^2)$. Thus Dijkstra's algorithm runs in time $O(n^2)$.

In fact, for an appropriate choice of z, the time is $\Omega(n^2)$ for K_n, the complete graph on n vertices, because every vertex is adjacent to every other. Thus the worst-case runtime is $\Theta(n^2)$.

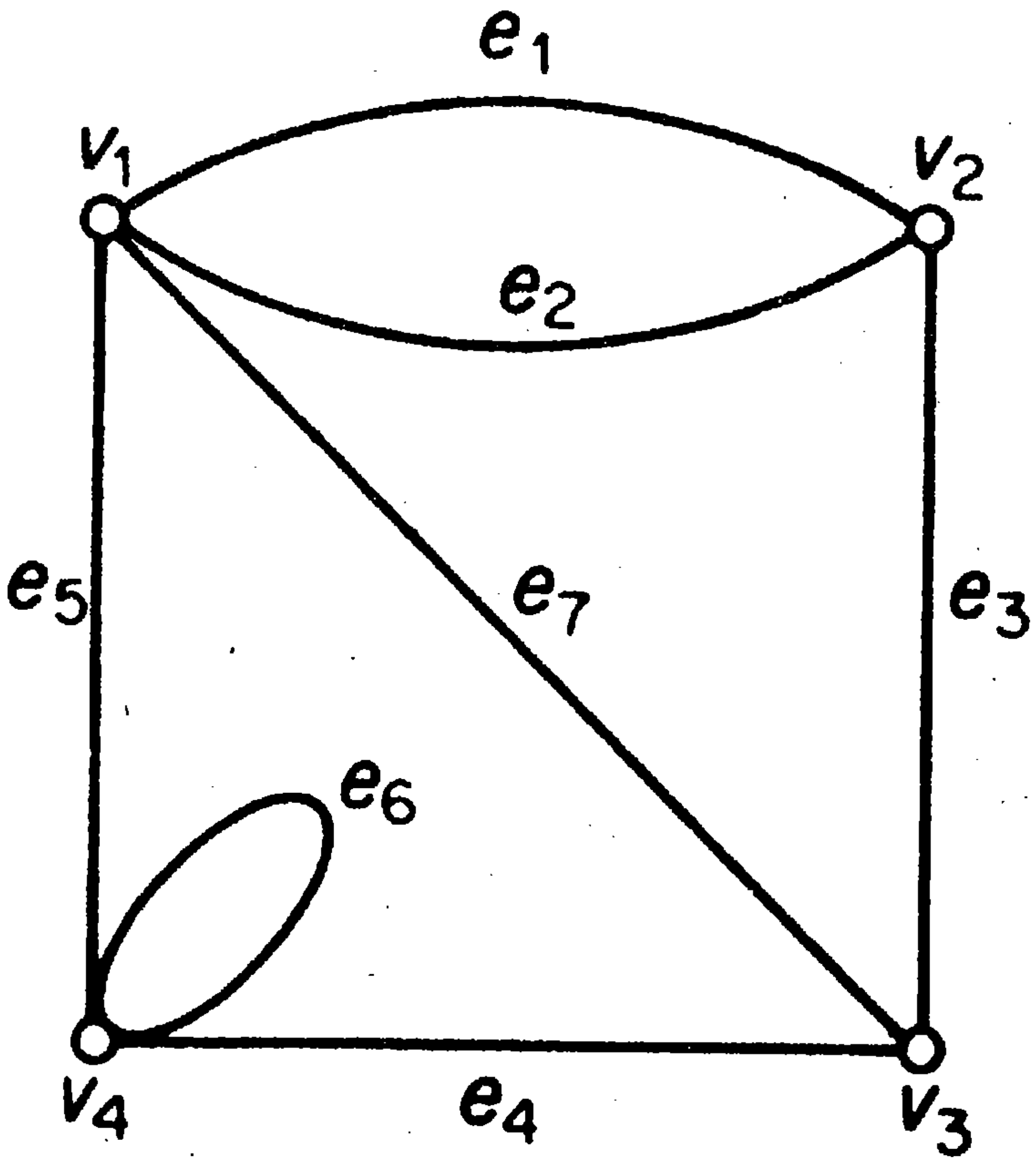
■

Any shortest-path algorithm that receives as input K_n, the complete graph on n vertices, must examine all of the edges of K_n at least once. Since K_n has $\frac{n(n-1)}{2}$ edges (see Exercise 16, Section 8.1), its worst-case run time must be at least $\frac{n(n-1)}{2} = \Omega(n^2)$. It follows from Theorem 8.4.5 that Algorithm 8.4.1 is **optimal**.

8.5 Representations of Graphs

THE INCIDENCE AND ADJACENCY MATRICES

To any graph G there corresponds a $\boldsymbol{m \times n}$ matrix called the incidence matrix of G. Let us denote the vertices of G by V_1, V_2, ... , V_m and the edges by e_1, e_2, · · · , e_n. Then the ***incidence matrix*** of G is the matrix M(G) = $[m_{ij}]$, where m_{ij} is the number of times (0, 1 or 2) that v_i and e_j are incident.



G

$$
\begin{array}{c|ccccccc}
 & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \\
\hline
v_1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
v_2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
v_3 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
v_4 & 0 & 0 & 0 & 1 & 1 & 2 & 0
\end{array}
$$

$\mathbf{M}(G)$

Another matrix associated with G is the ***adjacency matrix***; this is the $m \times n$ matrix $A(G) = [a_{ij}]$, in which a_{ij} is the number of edges joining V_i and V_j

$$
\begin{array}{c|cccc}
 & v_1 & v_2 & v_3 & v_4 \\
\hline
v_1 & 0 & 2 & 1 & 1 \\
v_2 & 2 & 0 & 1 & 0 \\
v_3 & 1 & 1 & 0 & 1 \\
v_4 & 1 & 0 & 1 & 1
\end{array}
$$

$\mathbf{A}(G)$

Problem 1.3.1:

Let M be the incidence matrix and A the adjacency matrix of a graph G.

(a) Show that every column sum of M is 2.

(b) What are the column sums of A?

Solution:

(a) Suppose we have n vertices and s edges and for $1 \leq j \leq s$, consider the column e_j. The only time 1 appears in the column e_j that vertex incidence on edge e_j. But each edge has exactly two vertices, thus in two places of column e_j , must appear the number 1. Therefore the sum of each column is equal 2.

(b) If we do not have loop, (since we consider two edges for loop) the column sums of A is equal to the number of edges which vertex v be incidence on them.

Problem 1.3.2:

Let G be bipartite. Show that the vertices of G can be enumerated so that the adjacency matrix of G has the form

$$\left[\begin{array}{c|c} 0 & A_{12} \\ \hline A_{21} & 0 \end{array}\right]$$

where A_{21} is the transpose of A_{12}•

Solution:

The graph G is bipartite, thus we partitioned vertices of V into two sets X and Y with m and n vertices respectively. Now we labeled again the vertices of V in such way that vertices in X labeled with $v_1 v_2 \cdots \ v_m$ and the vertices of Y labeled with $v_{m+1} v_{m+2} \cdots \ v_{m+n}$

$$
\begin{array}{c|c|c}
 & v_1 \ v_2 \ \cdots \ v_m & v_{m+1} \ v_{m+2} \cdots \ v_{m+n} \\
\hline
\begin{array}{c} v_1 \\ v_2 \\ \vdots \\ v_m \end{array} & \mathbf{0} & \boldsymbol{A}_{12} \\
\hline
\begin{array}{c} v_{m+1} \\ \vdots \\ v_{m+n} \end{array} & \boldsymbol{A}_{21} & \mathbf{0}
\end{array}
$$

8.6 Isomorphism of Graphs

The following instructions are given to two persons who cannot see each other's paper: "Draw and label five vertices *a*, *b*, *c*, *d*, and *e*. Connect *a* and *b*, *b* and *c*, *c* and *d*, *d* and *e*, and *a* and *e*." The graphs produced are shown in Figure 8.6.1. Surely these figures define the same graph even though they appear dissimilar. Such graphs are said to be **isomorphic.**

Graphs are isomorphic if they are the same as ***graphs,*** even though the names of the vertices and edges may be different.

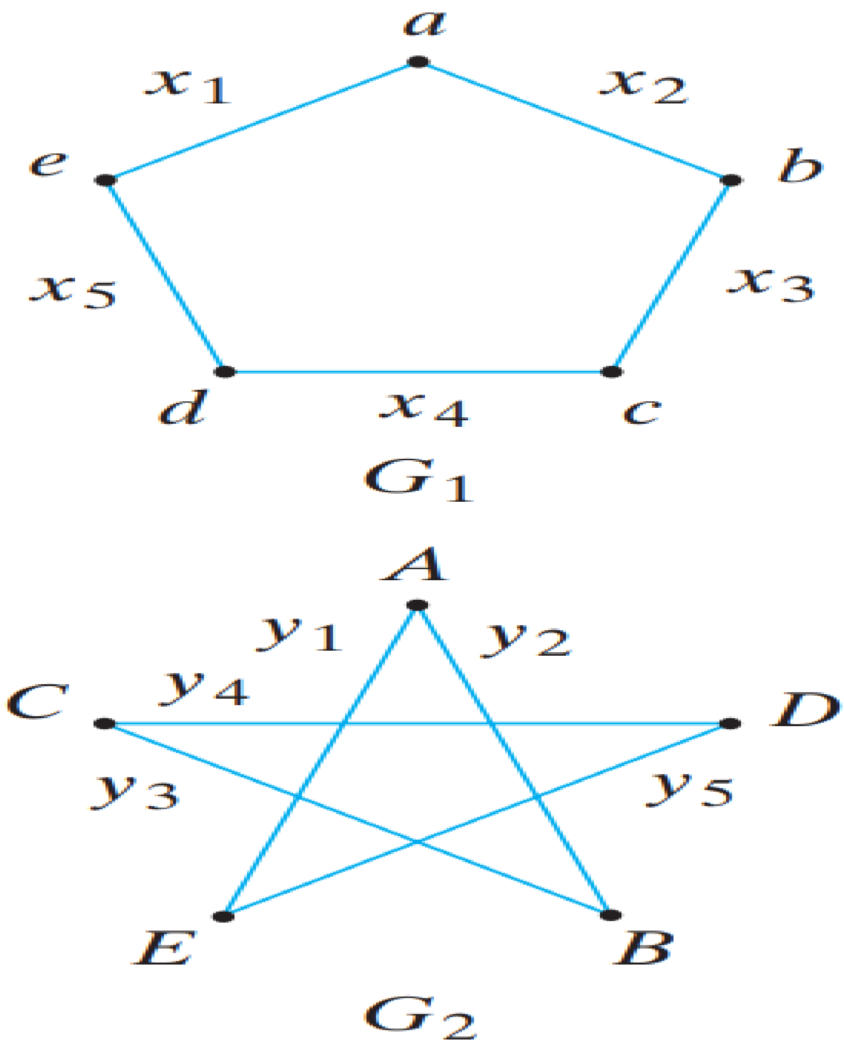


Figure 8.6.1
Isomorphic graphs.

There are many practical applications of graph isomorphism. Fingerprints can be modeled using graphs in which vertices represent local geometric patterns and edges represent relations between the patterns (see [Nandi]). In this way, a fingerprint graph can be potentially identified by checking it against a database of fingerprint graphs. A similar application of graph isomorphism represents chemical and biological structures as graphs and compares them to databases of such representations (see [Willett]). This latter work is particularly useful in pharmaceutical research.

Definition 8.6.1 Graphs G_1 and G_2 are ***isomorphic*** if there is a one-to-one, onto function f from the vertices of G_1 to the vertices of G_2 and a one-to-one, onto function g from the edges of G_1 to the edges of G_2, so that an edge e is incident on v and w in G_1 if and only if the edge $g(e)$ is incident on $f(v)$ and $f(w)$ in G_2. The pair of functions f and g is called an *isomorphism* of G_1 onto G_2.

Example 8.6.2 An isomorphism for the graphs G_1 and G_2 of Figure 8.6.1 is defined by

f (a) = A, f (b) = B, f (c) = C, f (d) = D, f (e) =E,

$$g(x_i) = y_i, \qquad i = 1, \ldots, 5.$$

We can think of functions f and g as "renaming functions."

Theorem 8.6.4

Graphs G_1 and G_2 are isomorphic if and only if for some ordering of their vertices, their adjacency matrices are equal.

Corollary 8.6.5 Let G_1 and G_2 be simple graphs. The following are equivalent:

(a) G_1 and G_2 are isomorphic.

(b) There is a one-to-one, onto function *f* from the vertex set of G_1 to the vertex set of G_2 satisfying the following: Vertices *v* and *w* are adjacent in G_1 if and only if the vertices *f (v)* and *f (w)* are adjacent in G_2.

Example 8.6.6 The adjacency matrix of graph G_1 in Figure 8.6.1 relative to the vertex ordering a, b, c, d, e,

$$
\begin{array}{c}
\\ a \\ b \\ c \\ d \\ e
\end{array}
\begin{array}{c}
\begin{array}{ccccc} a & b & c & d & e \end{array} \\
\begin{pmatrix}
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0
\end{pmatrix}
\end{array},
$$

is equal to the adjacency matrix of graph G2 in Figure 8.6.1 relative to the vertex ordering A, B, C, D, E,

$$
\begin{array}{c}
\\ A \\ B \\ C \\ D \\ E
\end{array}
\begin{array}{c}
\begin{array}{ccccc} A & B & C & D & E \end{array} \\
\begin{pmatrix}
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0
\end{pmatrix}
\end{array}.
$$

We see again that G_1 and G_2 are isomorphic. ■

The following is one way to show that two simple graphs G_1 and G_2 are not isomorphic. Find a property of G_1 that G_2 does not have but that G_2 would have if G_1 and G_2 were isomorphic. Such a property is called an **invariant**. More precisely, a property P is **an invariant** if whenever G_1 and G_2 are isomorphic graphs: If G_1 has property P, G_2 also has property P.

By Definition 8.6.1, if graphs G_1 and G_2 are isomorphic, there are one-to-one, onto functions from the edges (respectively, vertices) of G_1 to the edges (respectively, vertices) of G_2. Thus, if G_1 and G_2 are isomorphic, then G_1 and G_2 have the **same number of edges and the same number of vertices.** Therefore, if e and n are nonnegative integers, the properties “has e edges” and “has n vertices” are **invariants.**

Example 8.6.7

The graphs G_1 and G_2 in Figure 8.6.3 are not isomorphic, since G_1 has seven edges and G_2 has six edges and “has seven edges” is an **invariant**.

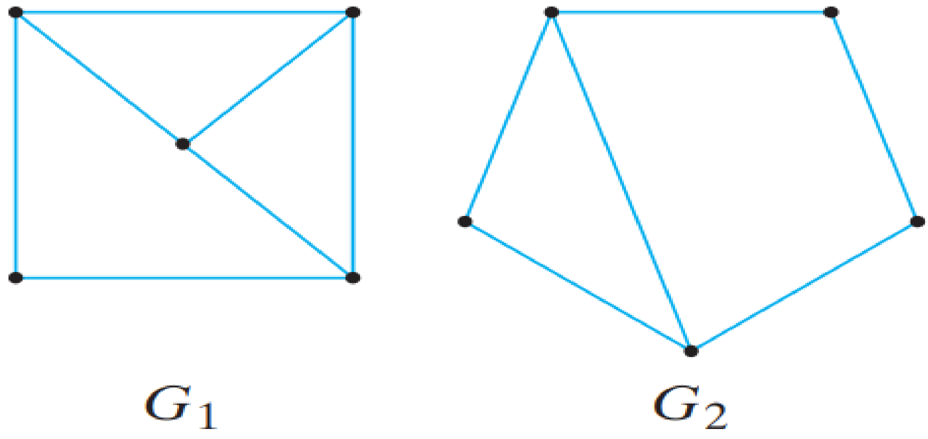


Figure 8.6.3 Nonisomorphic graphs. G_1 has seven edges and G_2 has six edges.

Example 8.6.9 Since "has a vertex of degree 3" is **an invariant**, the graphs G_1 and G_2 of Figure 8.6.4 are not isomorphic; G_1 has vertices (a and f) of degree 3, but G_2 does not have a vertex of degree 3. Notice that G_1 and G_2 have the same numbers of edges and vertices.

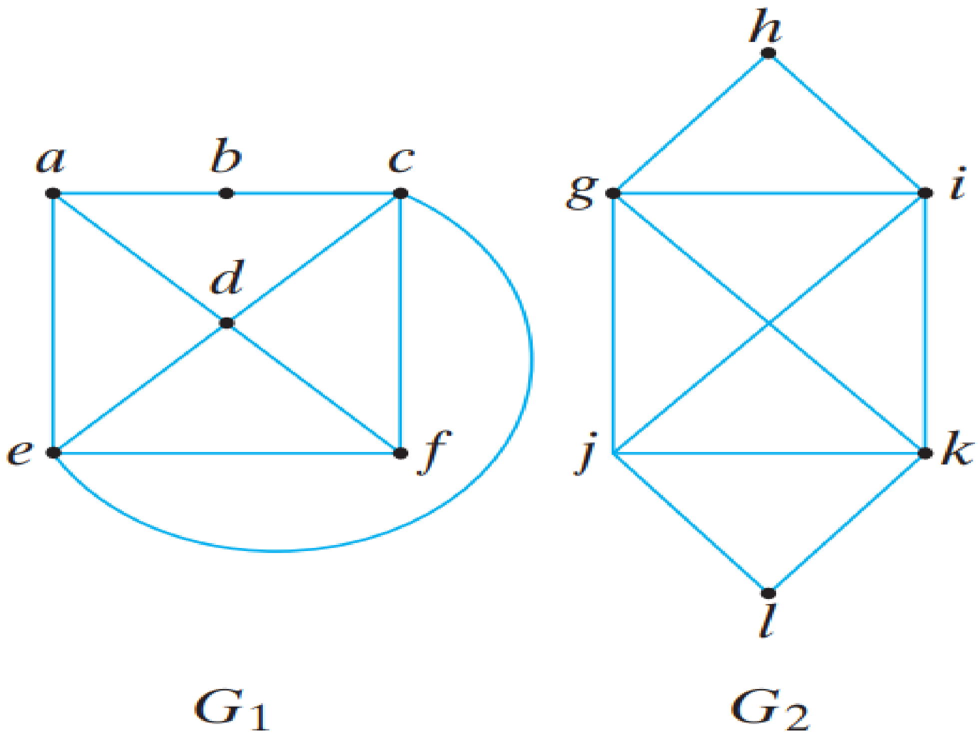


Figure 8.6.4 Nonisomorphic graphs. G_1 has vertices of degree 3, but G_2 has no vertices of degree 3.

Another **invariant** that is sometimes useful is "has a simple cycle of length *k*." We leave the proof that this property is an invariant to the exercises (Exercise 22).

Example 8.6.10 Since "has a simple cycle of length 3" is an invariant, the graphs G_1 and G_2 of Figure 8.6.5 are not isomorphic; the graph G_2 has a simple cycle of length 3, but all simple cycles in G_1 have length at least 4. Notice that G_1 and G_2 have the same numbers of edges and vertices and that every vertex in G_1 or G_2 has degree 4.

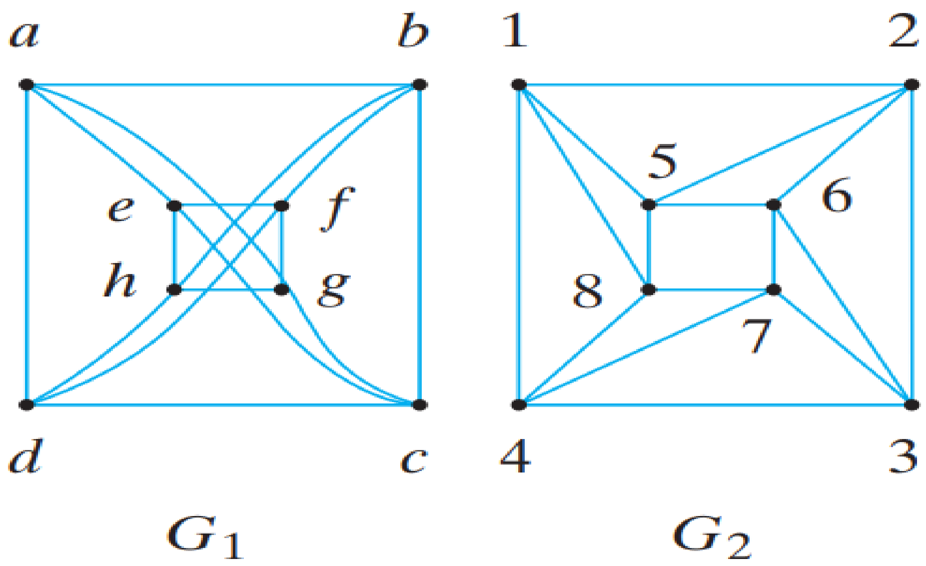


Figure 8.6.5 Nonisomorphic graphs. G_2 has a simple cycle of length 3, but G_1 has no simple cycles of length 3.

It would be easy to test whether a pair of graphs is isomorphic if we could find a small number of easily checked invariants that isomorphic graphs and *only* isomorphic graphs share. Unfortunately, no one has succeeded in finding such a set of invariants.

8.7 Planar Graphs

Three cities, C_1, C_2, and C_3, are to be directly connected by expressways to each of three other cities, C_4, C_5, and C_6. Can this road system be designed so that the expressways do not cross? A system in which the roads do cross is illustrated in Figure 8.7.1. If you try drawing a system in which the roads do not cross, you will soon be convinced that it cannot be done. Later in this section we explain carefully why it cannot be done.

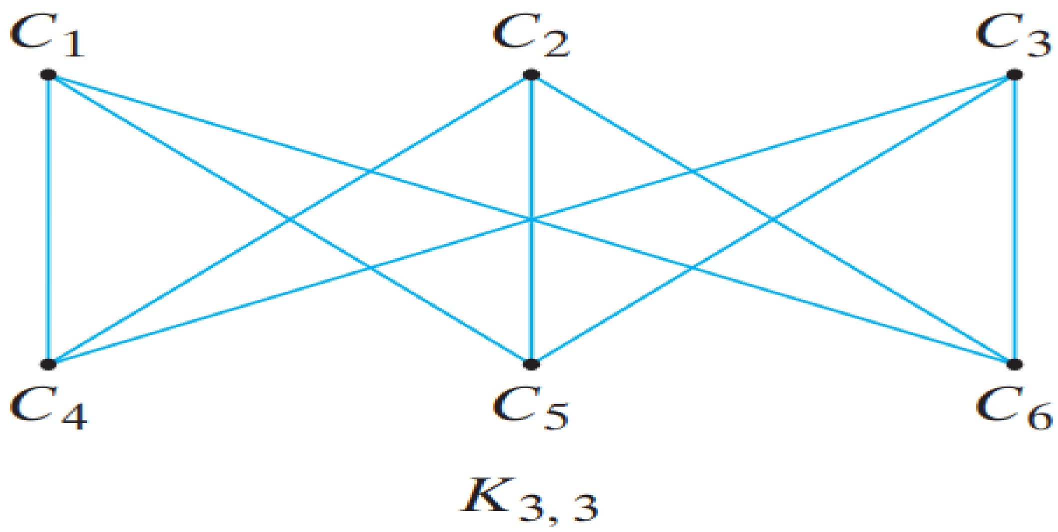


Figure 8.7.1 Cities connected by expressways.

Definition 8.7.1

A graph is *planar* if it can be drawn in the plane without its edges crossing.

In designing printed circuits it is desirable to have as few lines cross as possible; thus the designer of printed circuits faces the problem of planarity. If a connected, planar graph is drawn in the plane, the plane is divided into contiguous regions called **faces.** A face is characterized by the cycle that forms its boundary.

For example, in the graph of Figure 8.7.2, face *A* is bounded by the cycle (5, 2, 3, 4, 5) and face *C* is bounded by the cycle (1, 2, 5, 1). The outer face *D* is considered to be bounded by the cycle (1, 2, 3, 4, 6, 1).

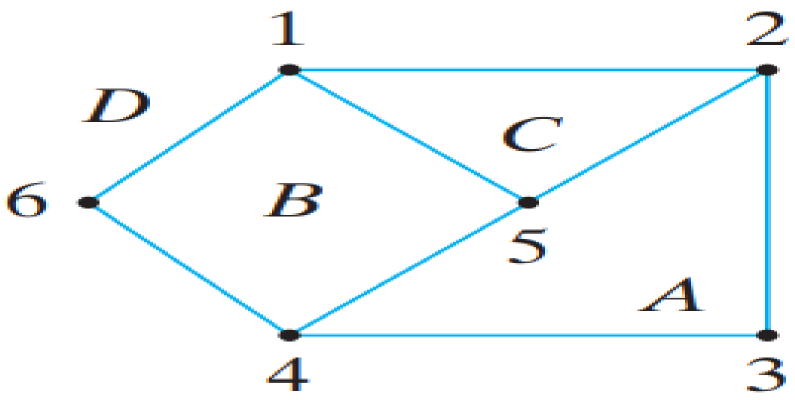


Figure 8.7.2 A connected, planar graph with $f = 4$ faces (A, B, C, D), $e = 8$ edges, and $v = 6$ vertices; $f = e - v + 2$.

The graph of Figure 8.7.2 has $f = 4$ faces, $e = 8$ edges, and $v = 6$ vertices. Notice that f, e, and v satisfy the equation

$$f = e - v + 2 \qquad (8.7.1)$$

In 1752, Euler proved that equation (8.7.1) holds for any connected planar graph. At the end of this section, we will show how to prove (8.7.1), but for now let us show how (8.7.1) can be used to show that certain graphs are not planar.

Example 8.7.2

Show that the graph $K_{3,3}$ of Figure 8.7.1 is not planar.

***SOLUTION*:** Suppose that $K_{3,3}$ is planar. Since every cycle has at least four edges, each face is bounded by at least four edges. Thus the number of edges that bound faces is at least $4f$. In a planar graph, each edge belongs to at most two bounding cycles. Therefore, $2e \geq 4f$. Using (8.7.1), we find that

$$2e \geq 4(e - v + 2) \qquad (8.7.2)$$

For the graph of Figure 8.7.1, $e = 9$ and $v = 6$, so (8.7.2) becomes

$$18 = 2 \cdot 9 \geq 4(9 - 6 + 2) = 20,$$

which is a contradiction. Therefore, $K_{3,3}$ is not planar.

■

By a similar kind of argument (see Exercise 15), we can show that the graph K_5 of Figure 8.7.3 is not planar.

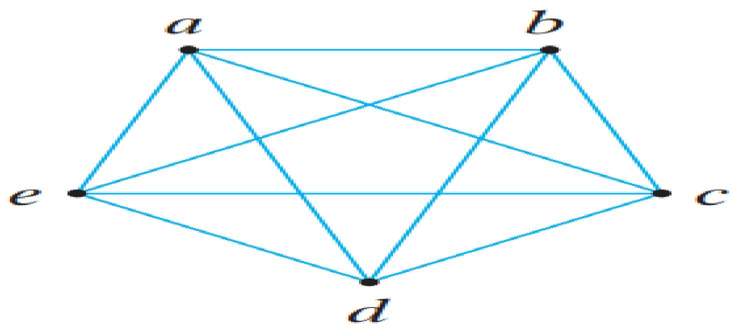


Figure 8.7.3 The nonplanar graph K_5.

Obviously, if a graph contains $K_{3,3}$ or K_5 as a subgraph, it cannot be planar. The converse is not true; however, if we introduce the concept of "homeomorphic graphs," we can obtain a true statement similar to the converse (see Theorem 8.7.7).

Definition 8.7.3 If a graph G has a vertex v of degree 2 and edges (v, v_1) and (v, v_2) with $v_1 \neq v_2$, we say that the edges (v, v_1) and (v, v_2) are in ***series***. A ***series reduction*** consists of deleting the vertex v from the graph G and replacing the edges (v, v_1) and (v, v_2) by the edge

(v_1, v_2). The resulting graph G' is said to be ***obtained from*** *G by a* ***series reduction***. By convention, *G* is said to be obtainable from itself by a series reduction.

Example 8.7.4 In the graph *G* of Figure 8.7.4, the edges (v, v_1) and (v, v_2) are in series. The graph G' of Figure 8.7.4 is obtained from *G* by a series reduction.

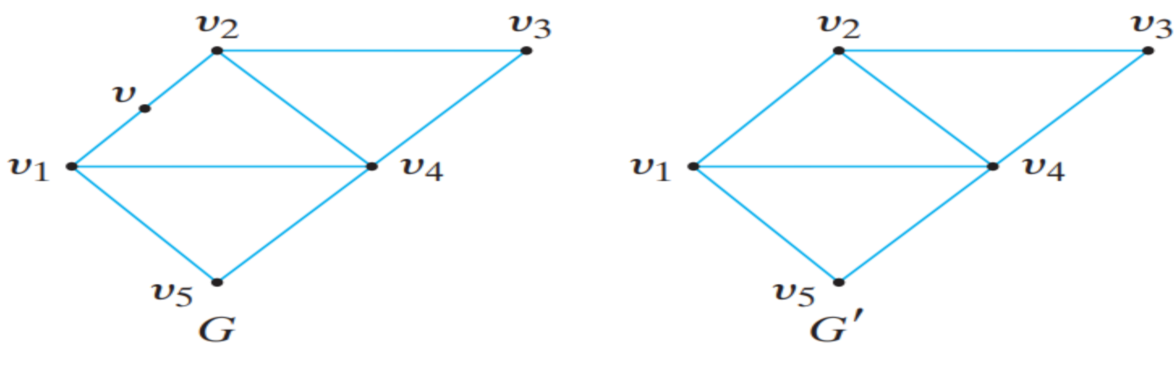


Figure 8.7.4 G' is obtained from G by a series reduction.

Definition 8.7.5 Graphs G_1 and G_2 are *homeomorphic* if G_1 and G_2 can be reduced to isomorphic graphs by performing a sequence of series reductions.

Example 8.7.6 The graphs G_1 and G_2 of Figure 8.7.5 are homeomorphic since they can both be reduced to the graph G' of Figure 8.7.5 by a sequence of series reductions.

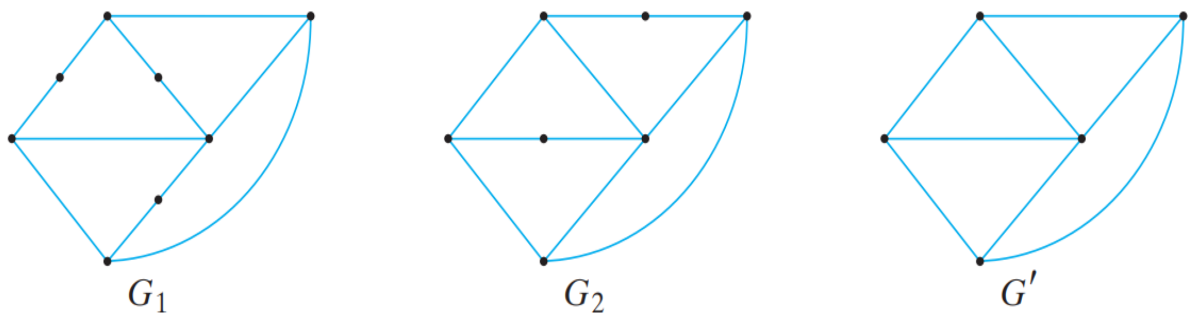


Figure 8.7.5 G_1 and G_2 are homeomorphic; each can be reduced to G'.

If we define a relation R on a set of graphs by the rule $G_1 R G_2$ if G_1 and G_2 are homeomorphic, R is an equivalence relation. Each equivalence class consists of a set of mutually homeomorphic graphs.

We now state a necessary and sufficient condition for a graph to be planar. The theorem was first stated and proved by Kuratowski in 1930. The proof appears in [Even 1979].

Theorem 8.7.7 Kuratowski's Theorem

A graph G is planar if and only if G does not contain a subgraph homeomorphic to K_5 or $K_{3,3}$.

Example 8.7.8 Show that the graph G of Figure 8.7.6 is not planar by using Kuratowski's Theorem.

SOLUTION Let us try to find $K_{3,3}$ in the graph G of Figure 8.7.6. We first note that the vertices a, b, f , and e each have degree 4. In $K_{3,3}$ each vertex has degree 3, so let us eliminate the edges *(a, b)* and *(f , e)* so that all vertices have degree 3 (see Figure 8.7.6).

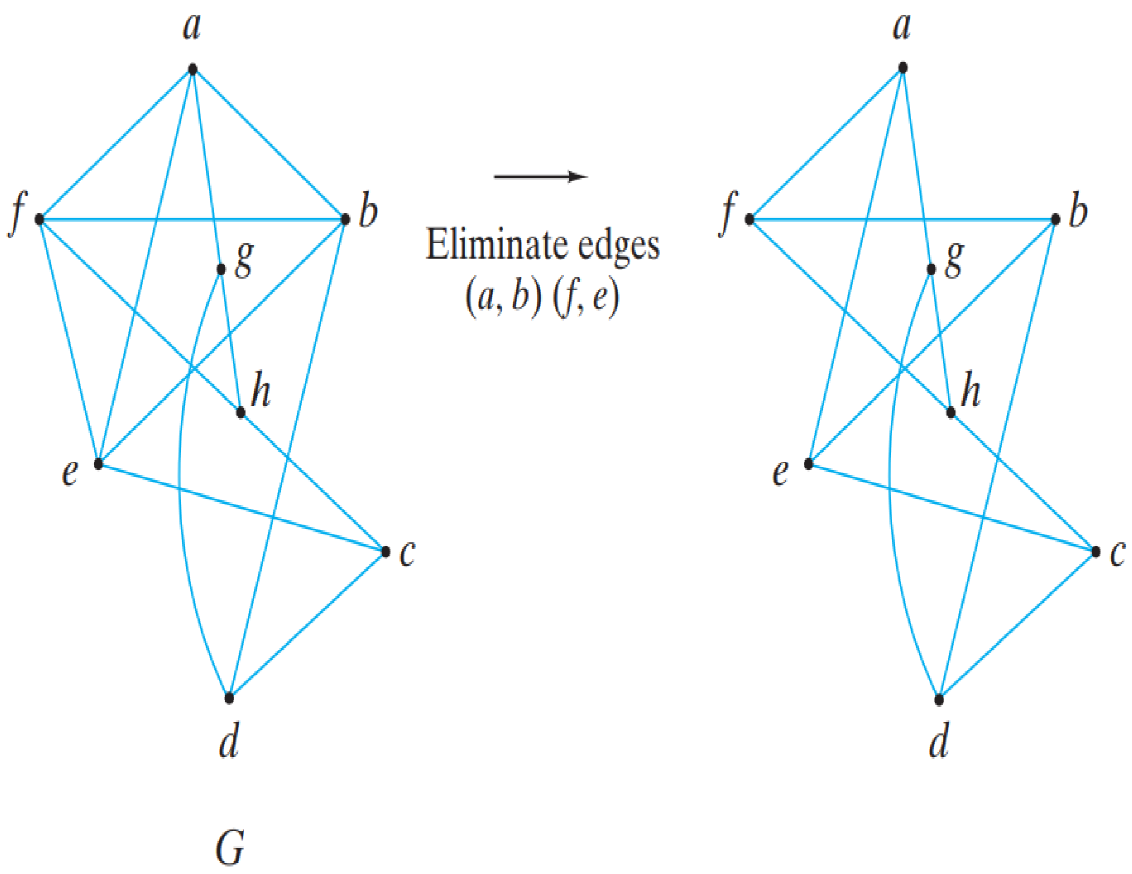


Figure 8.7.6 Eliminating edges to obtain a subgraph.

We note that if we eliminate one more edge, we will obtain two vertices of degree 2 and we can then carry out two series reductions. The resulting graph will have nine edges; since $K_{3,3}$ has nine edges, this approach looks promising. Using trial and error, we finally see that if we eliminate edge *(g, h)* and carry out the series reductions, we obtain an isomorphic copy of $K_{3,3}$ (see Figure 8.7.7).

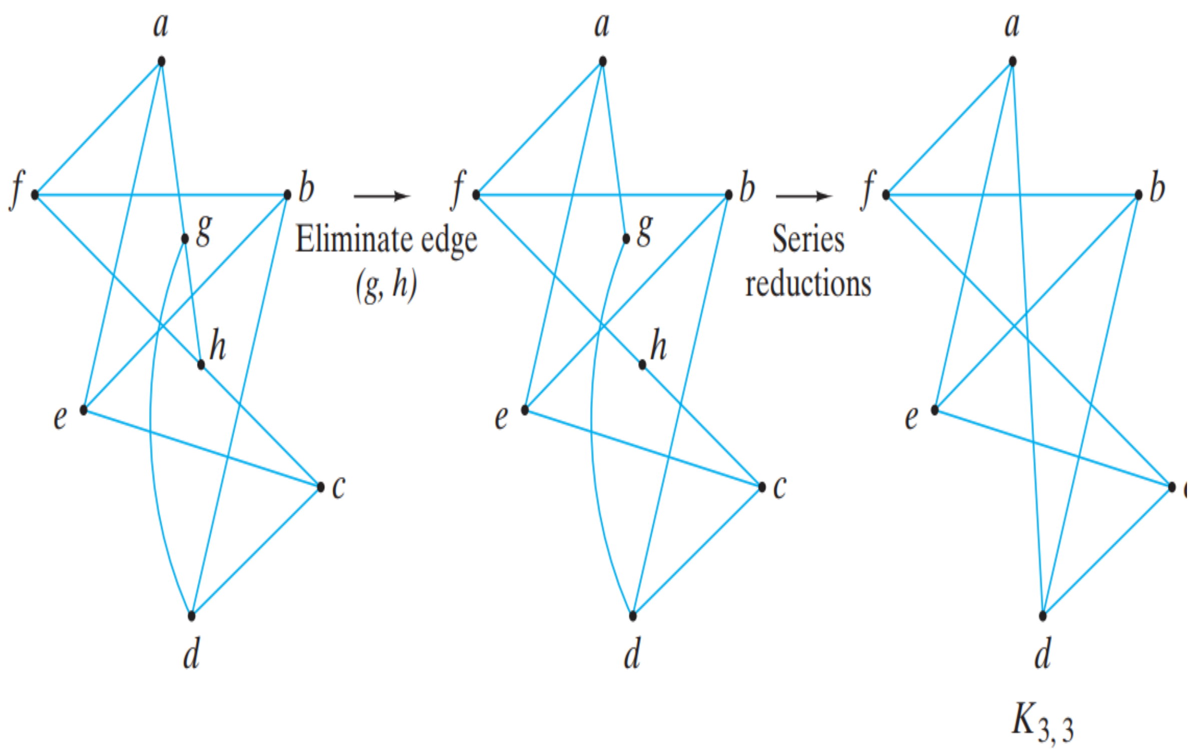


Figure 8.7.7 Elimination of an edge to obtain a subgraph, followed by series reductions.

Therefore, the graph G of Figure 8.7.6 is not planar, since it contains a subgraph homeomorphic to $K3,3$.

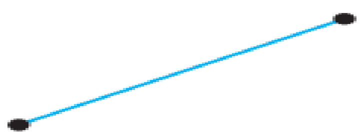
Theorem 8.7.9

Euler's Formula for Graphs

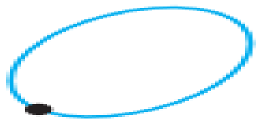
If G is a connected, planar graph with e edges, v vertices, and f faces, then

$$f = e - v + 2 \qquad (8.7.3)$$

Proof We will use induction on the number of edges.



$f = 1, e = 1, v = 2$



$f = 2, e = 1, v = 1$

Figure 8.7.8 The Basis Step of Theorem 8.7.9.

Suppose that $e = 1$. Then G is one of the two graphs shown in Figure 8.7.8. In either case, the formula holds. We have verified the Basis Step. Suppose that the formula holds for connected, planar graphs with n edges. Let G be a graph with $n+1$ edges. First, suppose that G contains no cycles. Pick a vertex v and trace a path starting at v. Since G is cycle-free, every time we trace an edge, we arrive at a new vertex. Eventually, we will reach a vertex *a*, with degree 1, that we cannot leave (see Figure 8.7.9).

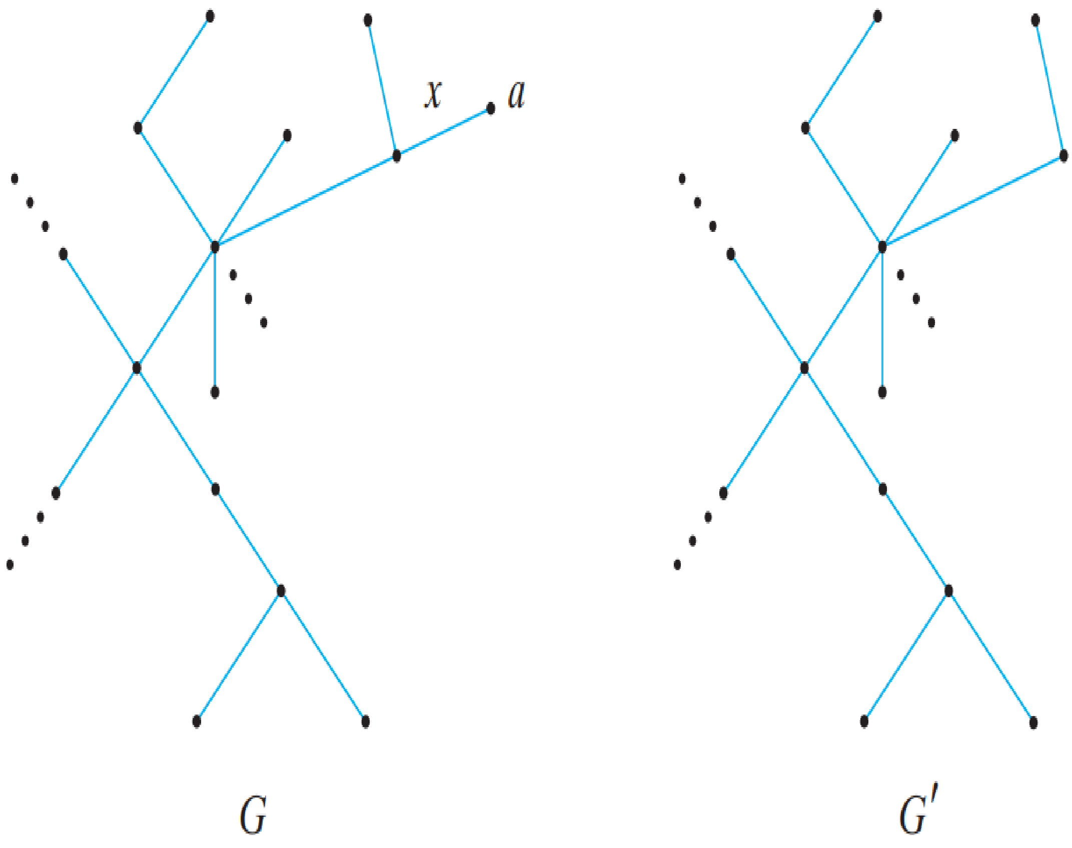


Figure 8.7.9 The proof of Theorem 8.7.9 for the case that G has no cycles. We find a vertex a of degree 1 and delete a and the edge x incident on it.

We delete *a* and the edge *x* incident on *a* from the graph *G*. The resulting graph

G' has n edges; hence, by the inductive assumption, (8.7.3) holds for G'. Since G has one more edge than G', one more vertex than G', and the same number of faces as G', it follows that (8.7.3) also holds for G.

Now suppose that G contains a cycle. Let x be an edge in a cycle (see Figure 8.7.10).

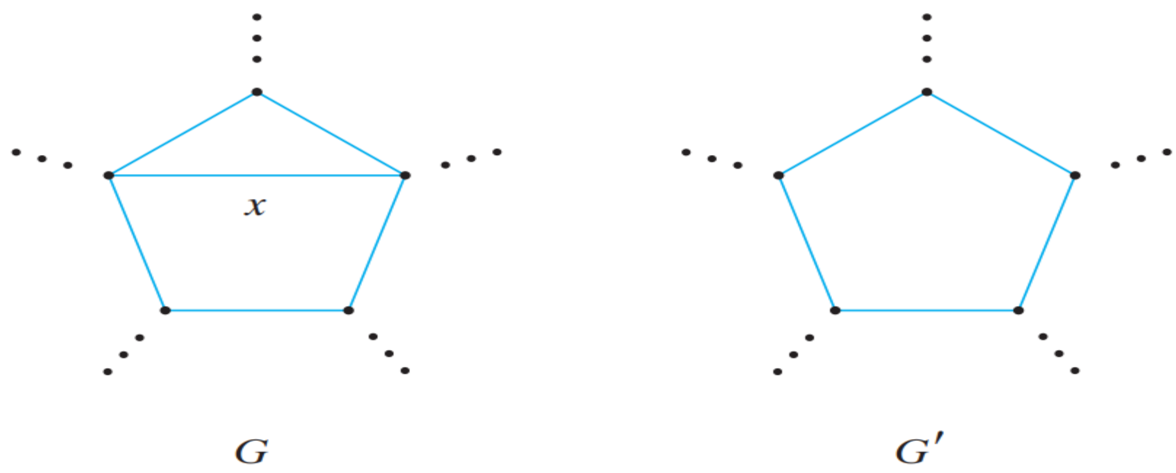


Figure 8.7.10 The proof of Theorem 8.7.9 for the case that G has a cycle. We delete edge x in a cycle.

Now x is part of a boundary for two faces. This time we delete the edge x but no vertices to obtain the graph G' (see Figure 8.7.10). Again G' has n edges; hence, by the inductive assumption, (8.7.3) holds for G'. Since G has one more face than G', one more edge than G', and the same number of vertices as G', it follows that (8.7.3) also holds for G.

■