CHAPTER 13

Graphs

Data Abstraction and Problem Solving with JAVA: Walls and Mirrors

Carrano / Prichard

Figure 13.1

An ordinary line graph

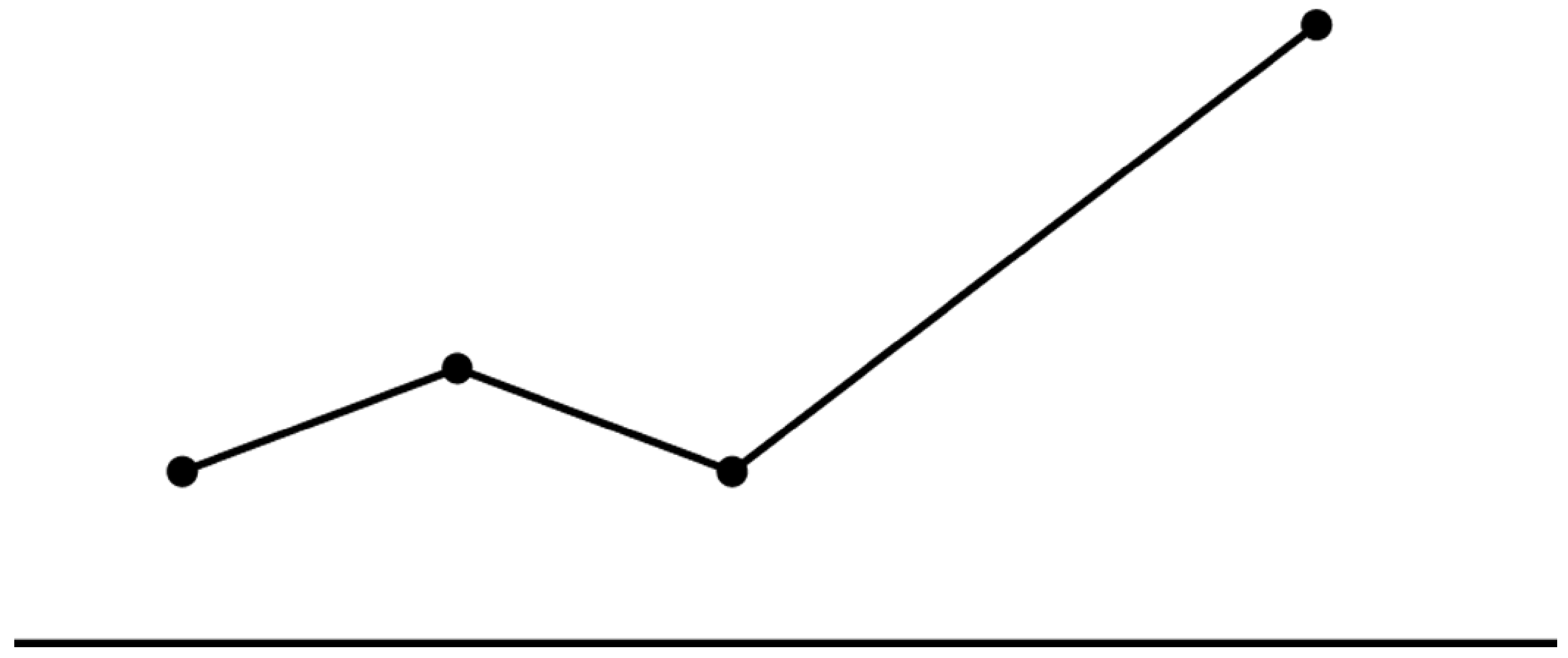
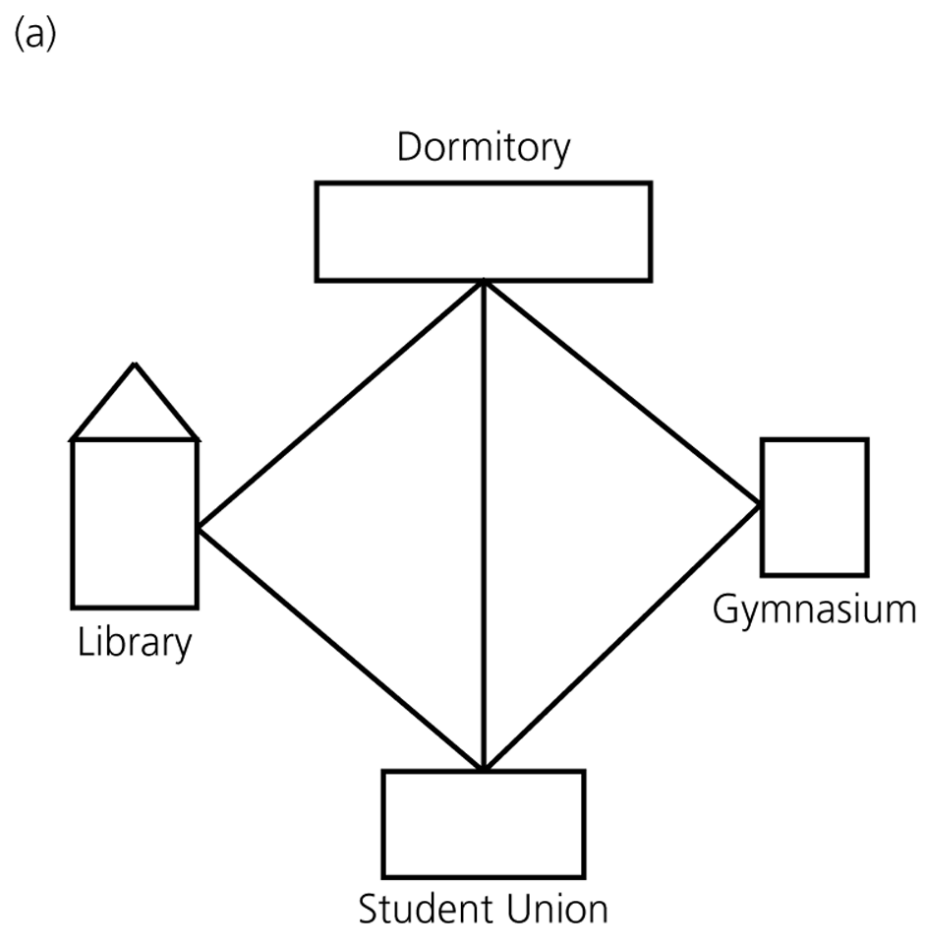


Figure 13.2

a) A campus map as a graph; b) a subgraph



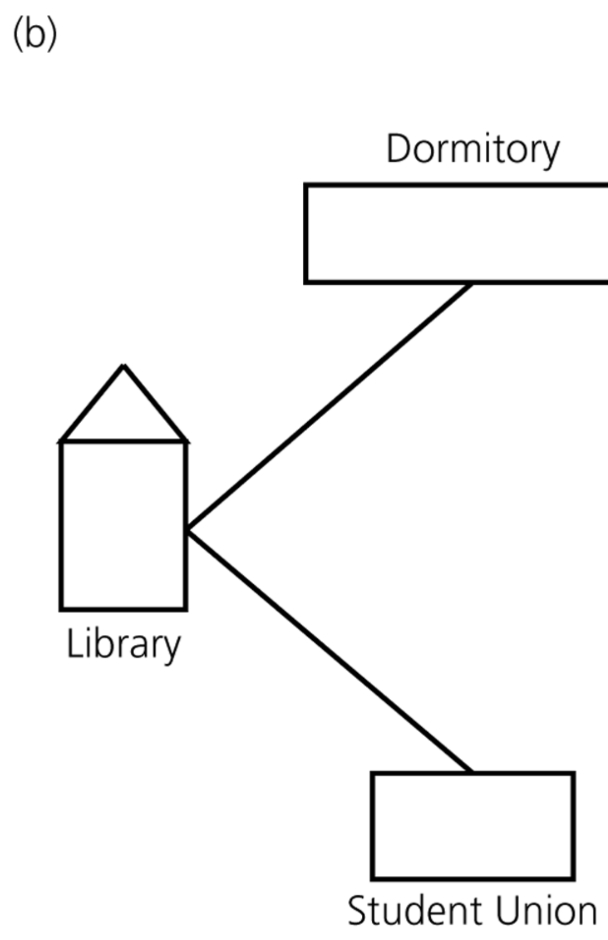


Figure 13.3

Graphs that are a) connected; b) disconnected; and c) complete

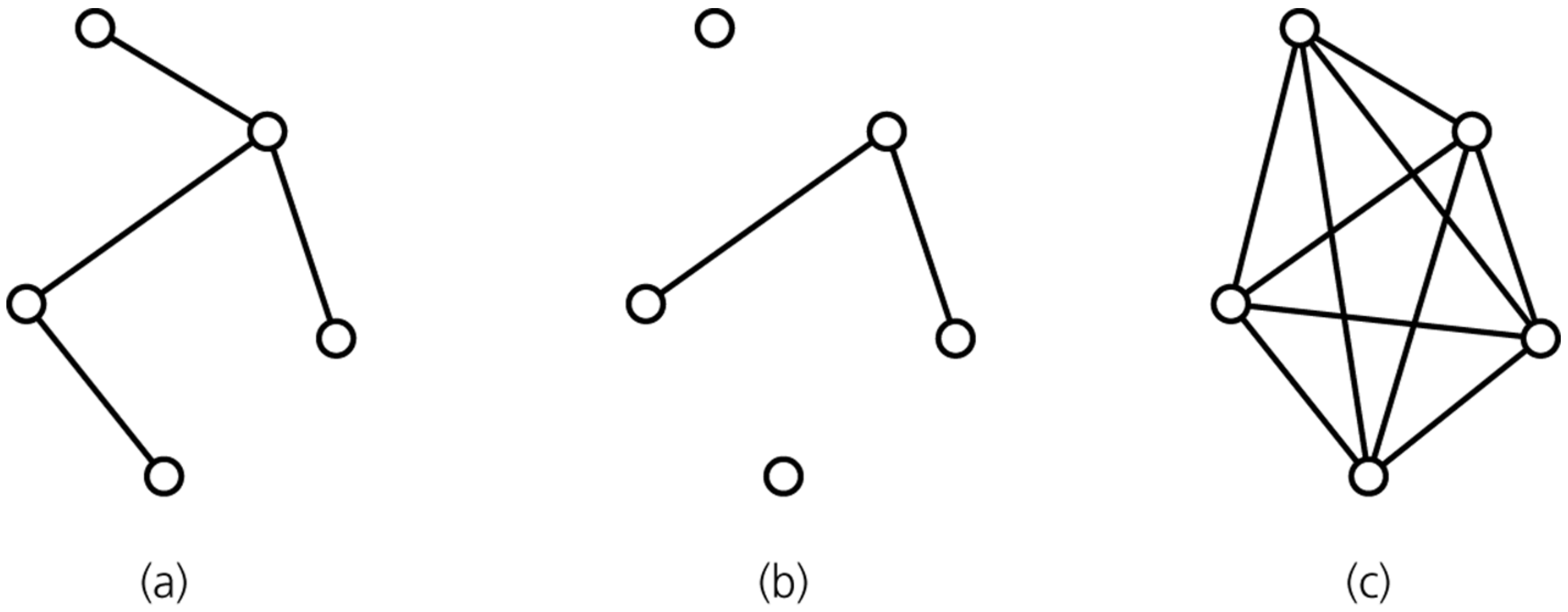
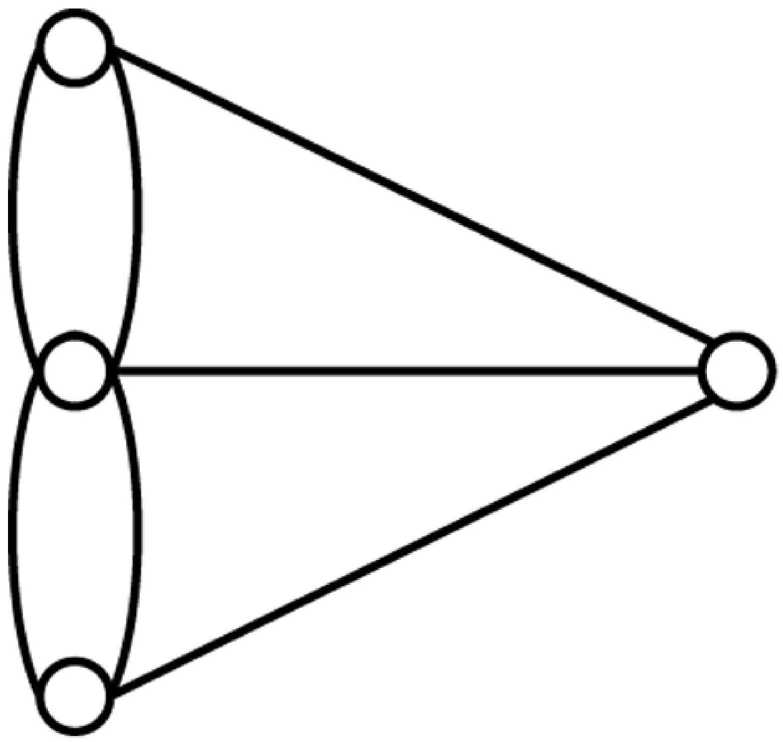
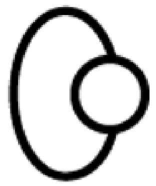


Figure 13.4

a) A multigraph is not a graph; b) a self edge is not allowed in a graph



(a)



(b)

Figure 13.5a

a) A weighted graph

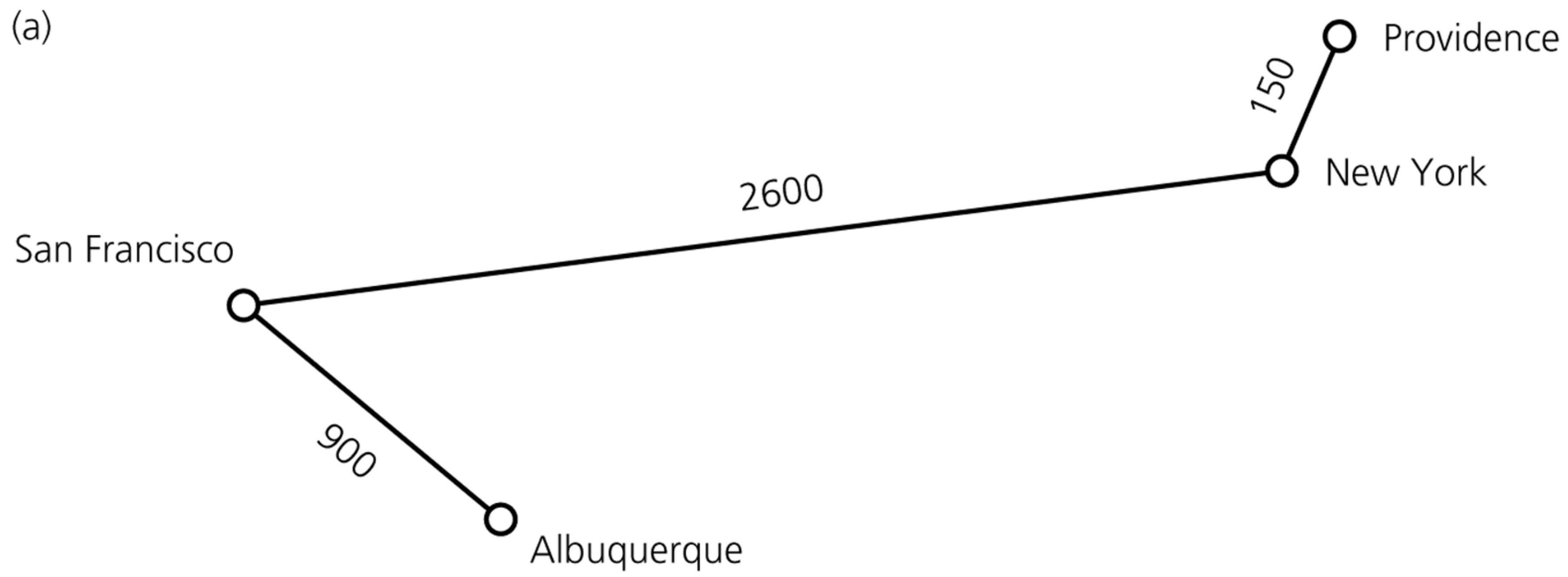


Figure 13.5b

b) A directed graph

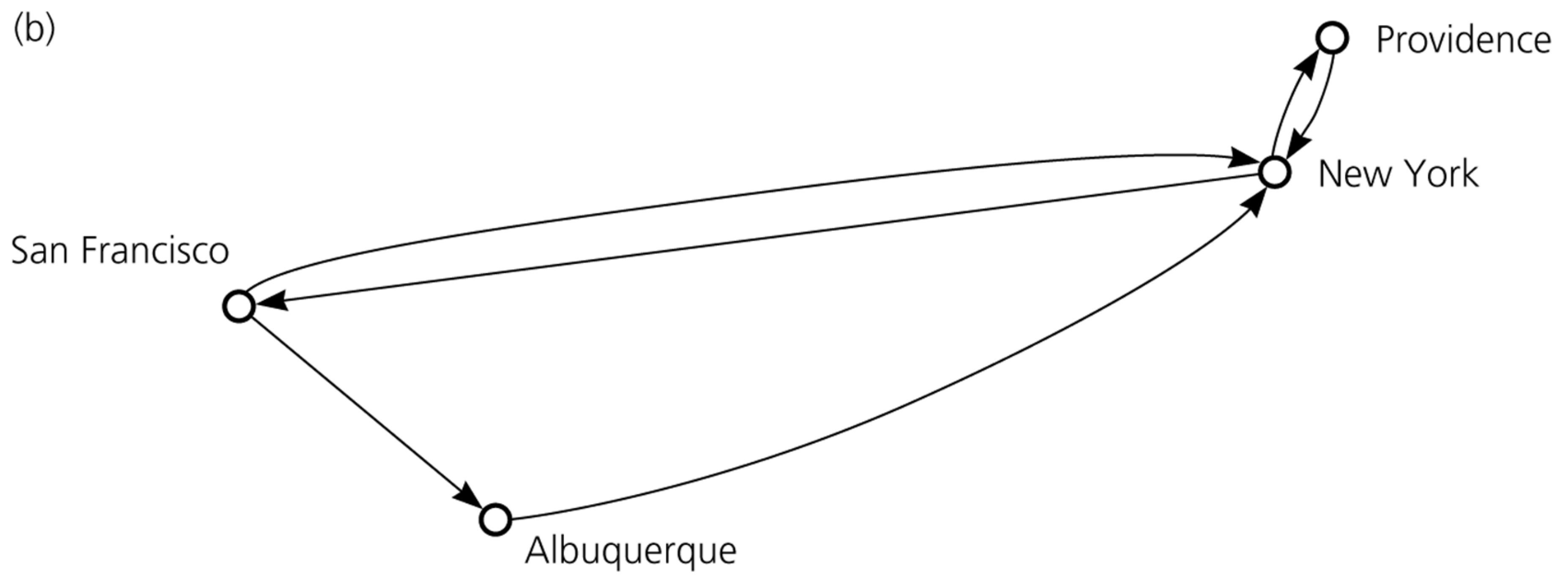
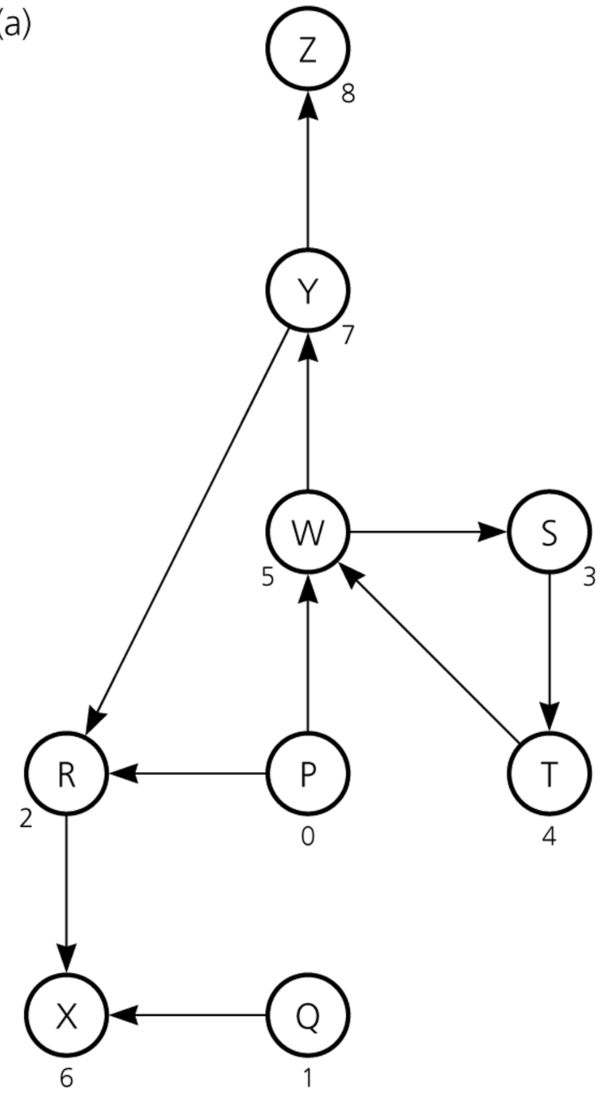


Figure 13.6

a) A directed graph and b) its adjacency matrix

(a)



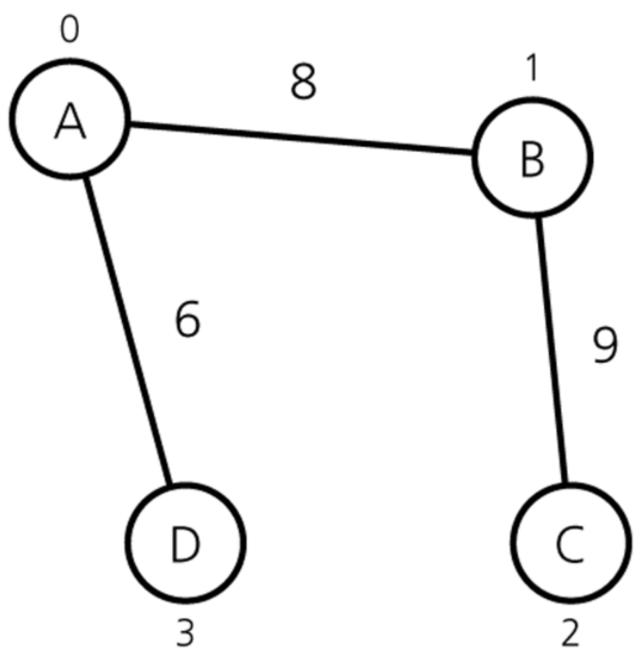
(b)

		0	1	2	3	4	5	6	7	8
		P	Q	R	S	T	W	X	Y	Z
0	P	0	0	1	0	0	1	0	0	0
1	Q	0	0	0	0	0	0	1	0	0
2	R	0	0	0	0	0	0	1	0	0
3	S	0	0	0	0	1	0	0	0	0
4	T	0	0	0	0	0	1	0	0	0
5	W	0	0	0	1	0	0	0	1	0
6	X	0	0	0	0	0	0	0	0	0
7	Y	0	0	1	0	0	0	0	0	1
8	Z	0	0	0	0	0	0	0	0	0

Figure 13.7

a) A weighted undirected graph and b) its adjacency matrix

(a)

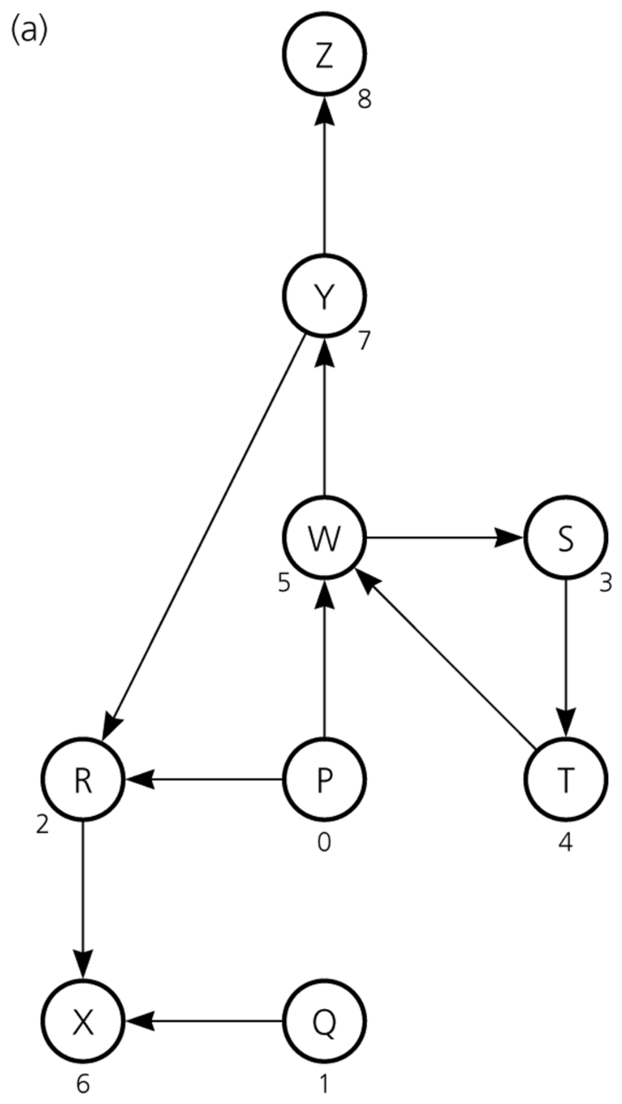


(b)

$$
\begin{array}{cc|cccc}
 & & 0 & 1 & 2 & 3 \\
 & & A & B & C & D \\
\hline
0 & A & \infty & 8 & \infty & 6 \\
1 & B & 8 & \infty & 9 & \infty \\
2 & C & \infty & 9 & \infty & \infty \\
3 & D & 6 & \infty & \infty & \infty
\end{array}
$$

Figure 13.8

a) A directed graph and b) its adjacency list



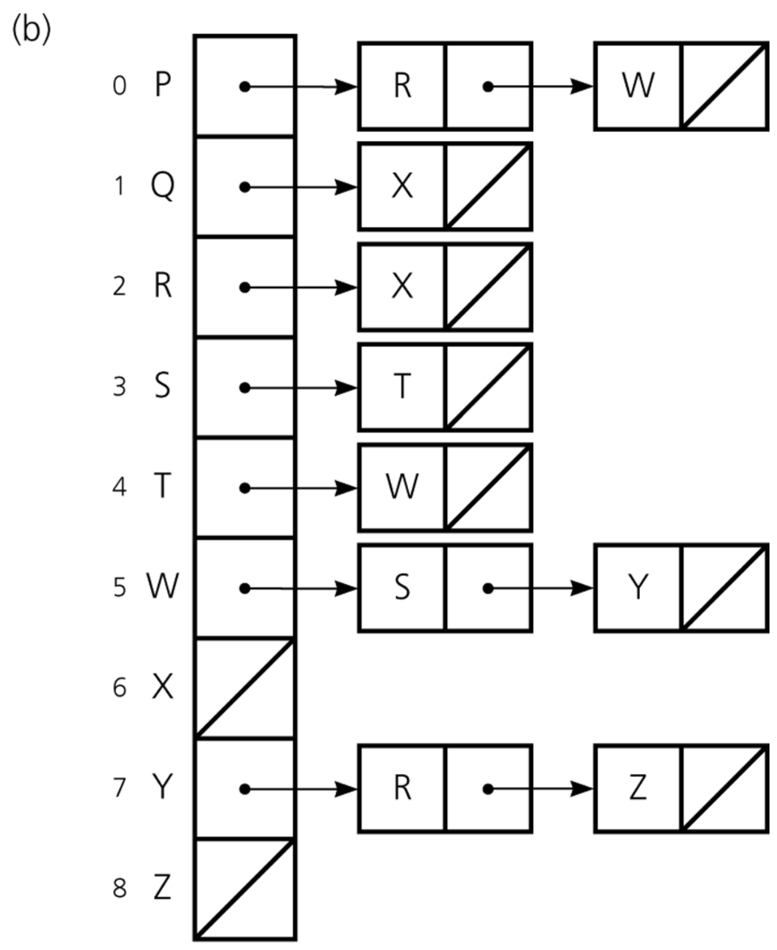


Figure 13.9

a) A weighted undirected graph and b) its adjacency list

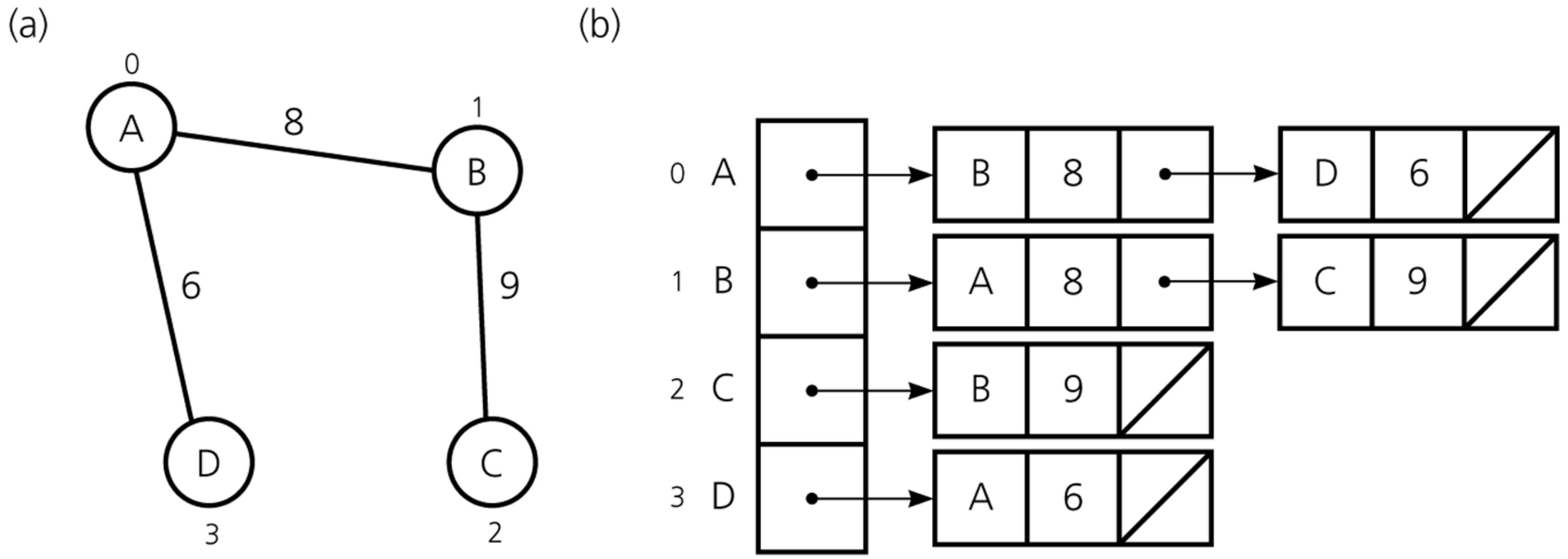


Figure 13.10

Visitation order for a) a depth-first search; b) a breadth-first search

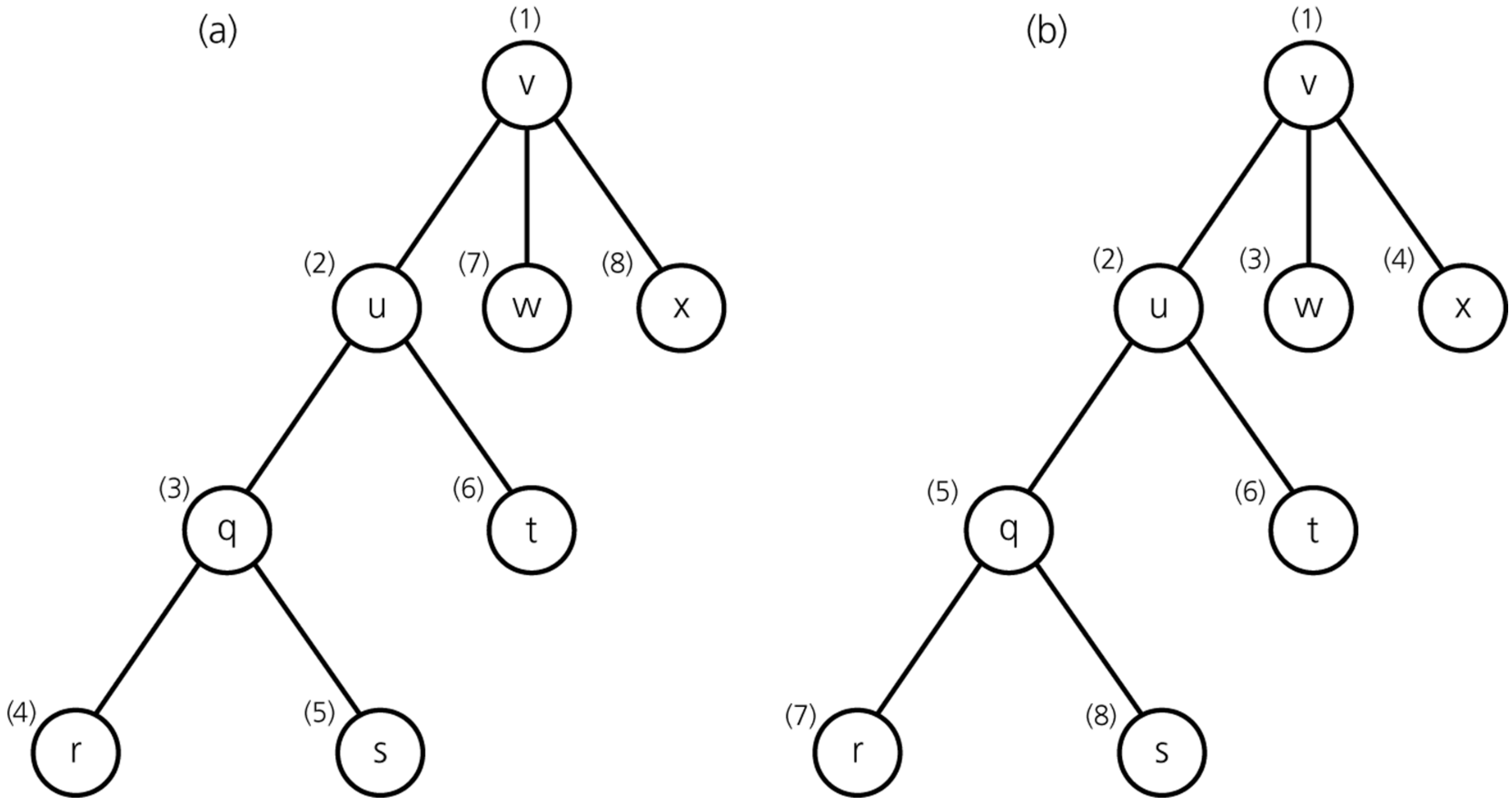


Figure 13.11

A connected graph with cycles

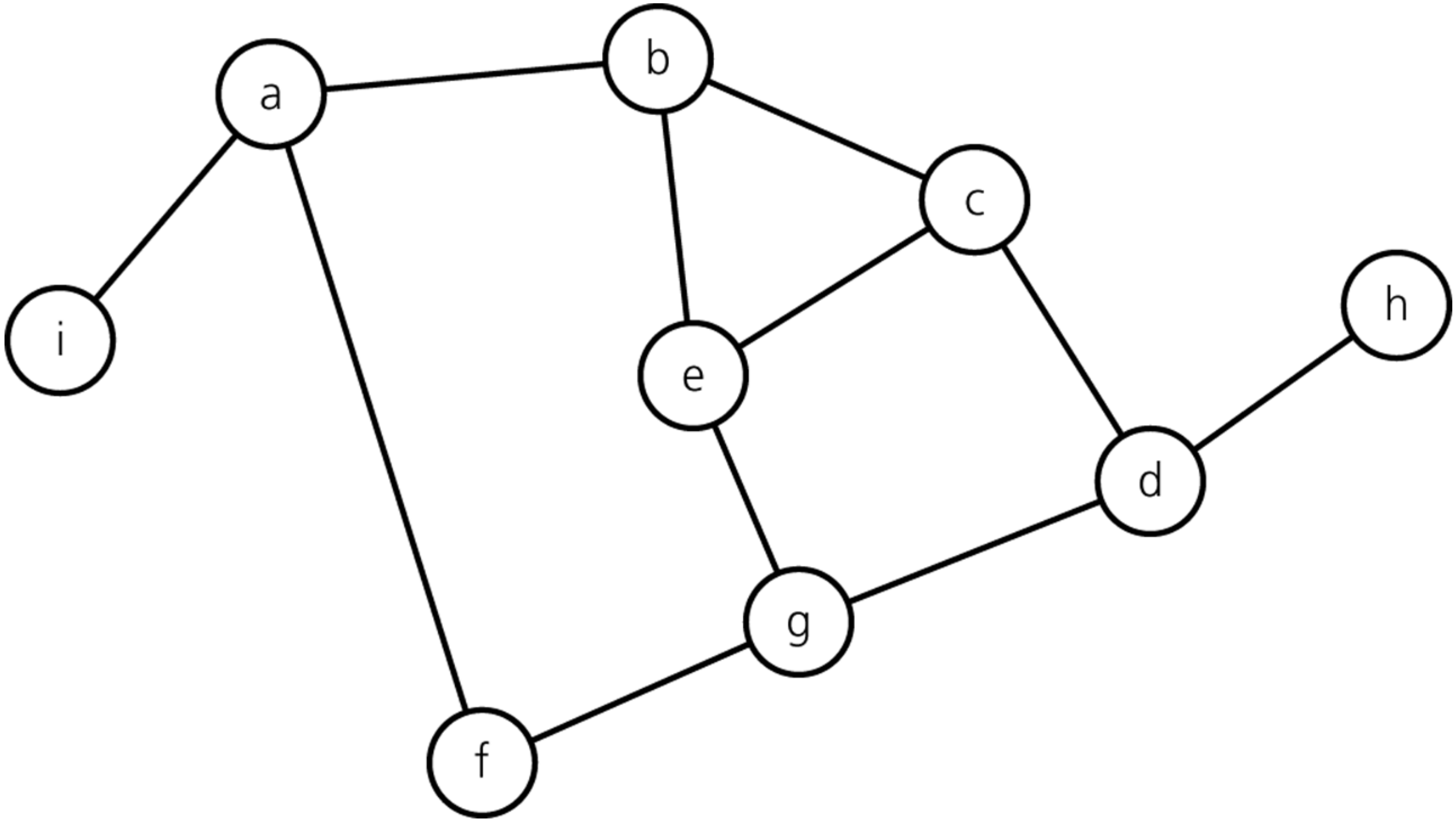


Figure 13.12

The results of a depth-first traversal, beginning at vertex *a*, of the graph in Figure 13-11

Node visited	Stack (bottom to top)
a	a
b	a b
c	a b c
d	a b c d
g	a b c d g
e	a b c d g e
(backtrack)	a b c d g
f	a b c d g f
(backtrack)	a b c d g
(backtrack)	a b c d
h	a b c d h
(backtrack)	a b c d
(backtrack)	a b c
(backtrack)	a b
(backtrack)	a
i	a i
(backtrack)	a
(backtrack)	*(empty)*

Figure 13.13

The results of a breadth-first traversal, beginning at vertex *a*, of the graph in Figure 13-11

Node visited	Queue (front to back)
a	a
	(empty)
b	b
f	b f
i	b f i
	f i
c	f i c
e	f i c e
	i c e
g	i c e g
	c e g
	e g
d	e g d
	g d
	d
	(empty)
h	h
	(empty)

Figure 13.14

A directed graph without cycles

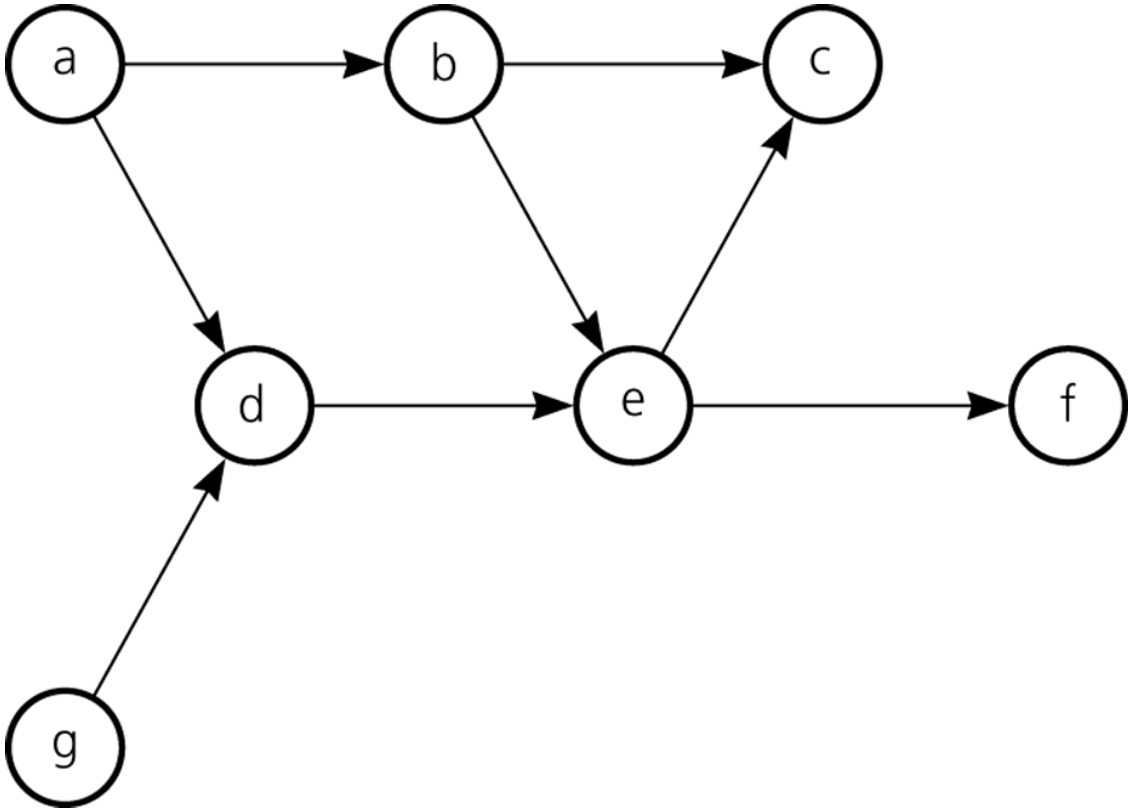


Figure 13.15

The graph in Figure 13-14 arranged according to the topological orders a) *a, g, d, b, e, c, f* and b) *a, b, g, d, e, f, c*

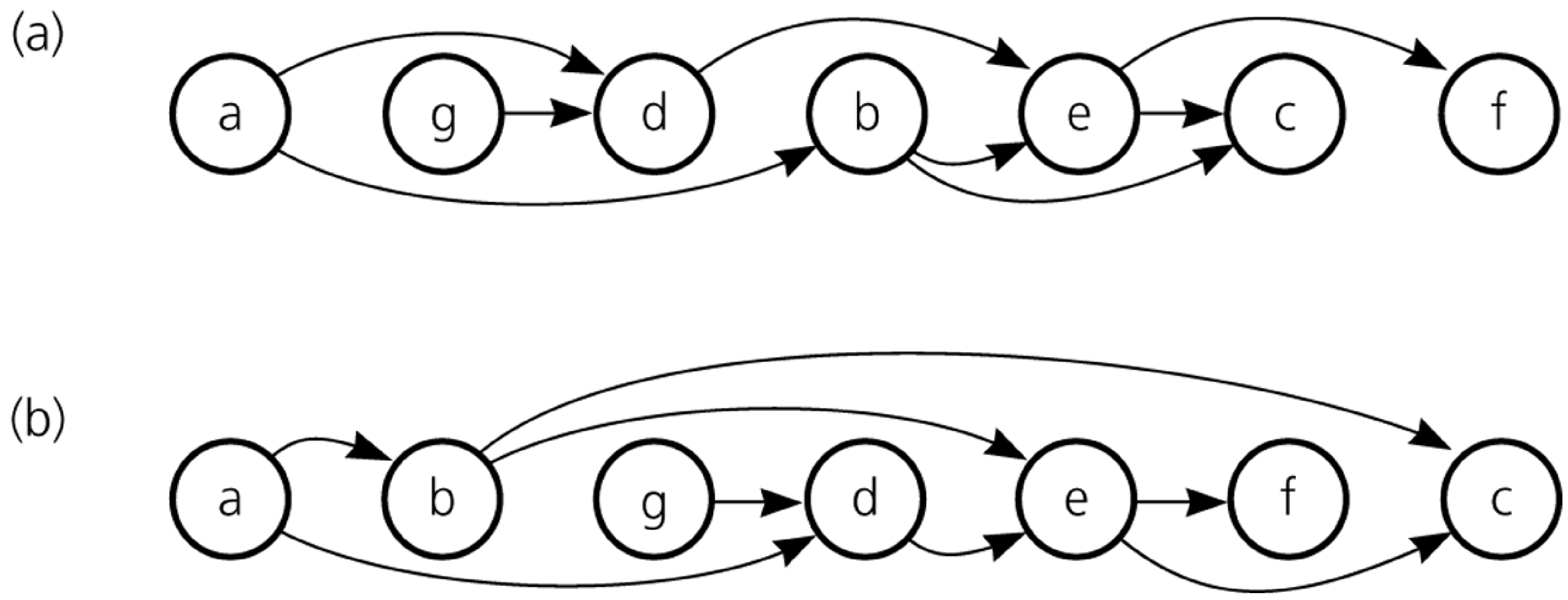
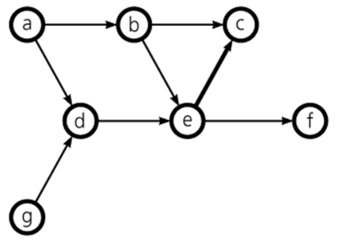


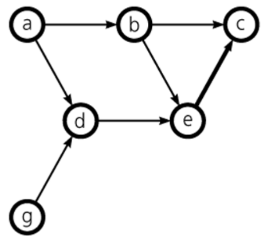
Figure 13.16a

A trace of ***`topSort1`*** for the graph in Figure 13-14

Graph **`theGraph`** List **`aList`**



Remove f from **`theGraph`**;
add it to **`aList`**

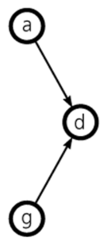


f

Remove c from **`theGraph`**;
add it to **`aList`**

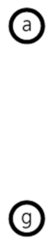
Graph **`theGraph`** List **`aList`**

Remove b from **`theGraph`**;
add it to **`aList`**



b e c f

Remove d from **`theGraph`**;
add it to **`aList`**



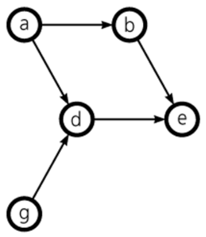
d b e c f

Remove g from **`theGraph`**;
add it to **`aList`**

Figure 13.16b

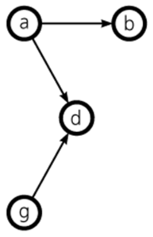
A trace of ***`topSort1`*** for the graph in Figure 13-14

Graph **`theGraph`** List **`aList`**



c f

Remove e from **`theGraph`**;
add it to **`aList`**



e c f

Graph **`theGraph`** List **`aList`**

g d b e c f

Remove a from **`theGraph`**;
add it to **`aList`**

a g d b e c f

Figure 13.17

A trace of ***topSort2*** for the graph in Figure 13-14

Action	Stack **s** (bottom to top)	List **aList** (beginning to end)
Push a	a	
Push g	a g	
Push d	a g d	
Push e	a g d e	
Push c	a g d e c	
Pop c, add c to **aList**	a g d e	c
Push f	a g d e f	c
Pop f, add f to **aList**	a g d e	f c
Pop e, add e to **aList**	a g d	e f c
Pop d, add d to **aList**	a g	d e f c
Pop g, add g to **aList**	a	g d e f c
Push b	a b	g d e f c
Pop b, add b to **aList**	a	b g d e f c
Pop a, add a to **aList**	*(empty)*	a b g d e f c

Figure 13.18

A spanning tree for the graph in Figure 13-11

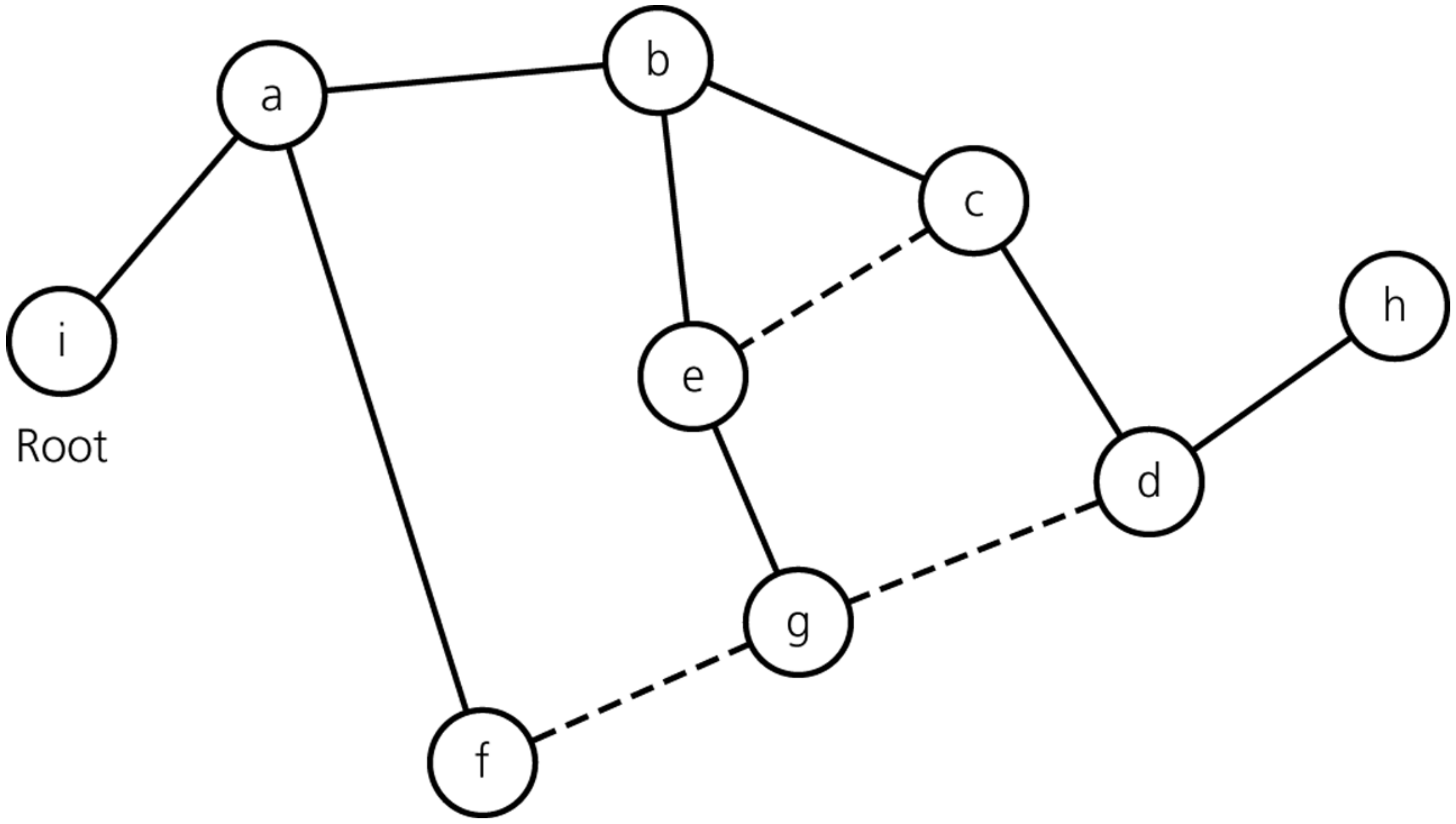


Figure 13.19

Connected graphs that each have four vertices and three edges

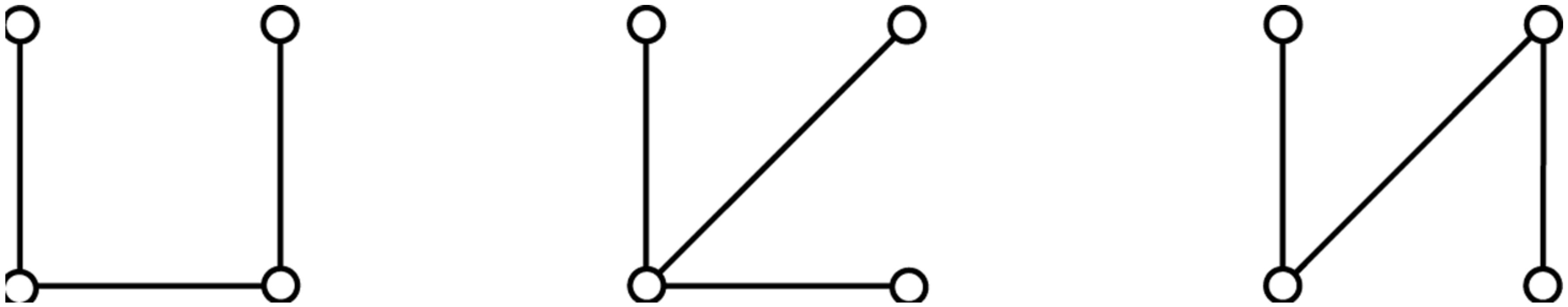
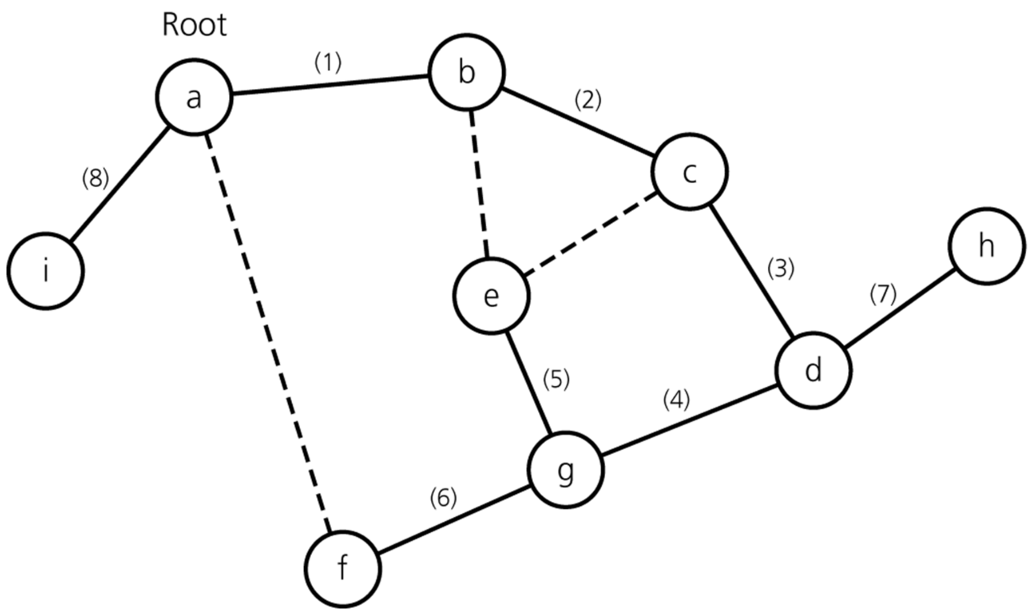


Figure 13.20

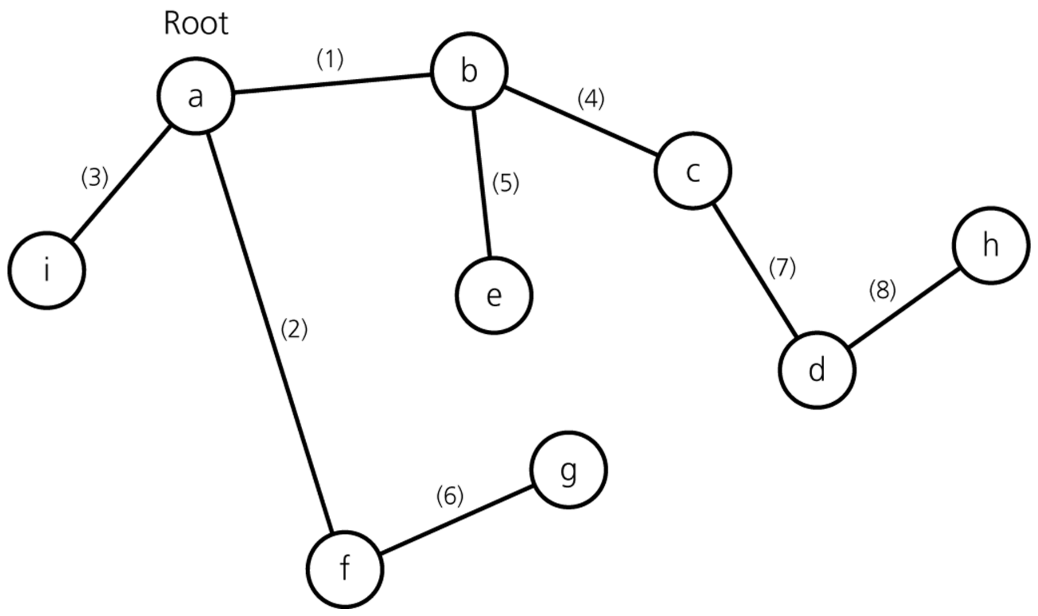
The DFS spanning tree rooted at vertex *a* for the graph in Figure 13-11



The DFS spanning tree algorithm visits vertices in this order: a, b, c, d, g, e, f, h, i. Numbers indicate the order in which the algorithm marks edges.

Figure 13.21

The BFS spanning tree rooted at vertex *a* for the graph in Figure 13-11



The BFS spanning tree algorithm visits vertices in this order: a, b, f, i, c, e, g, d, h. Numbers indicate the order in which the algorithm marks edges.

Figure 13.22

A weighted, connected, undirected graph

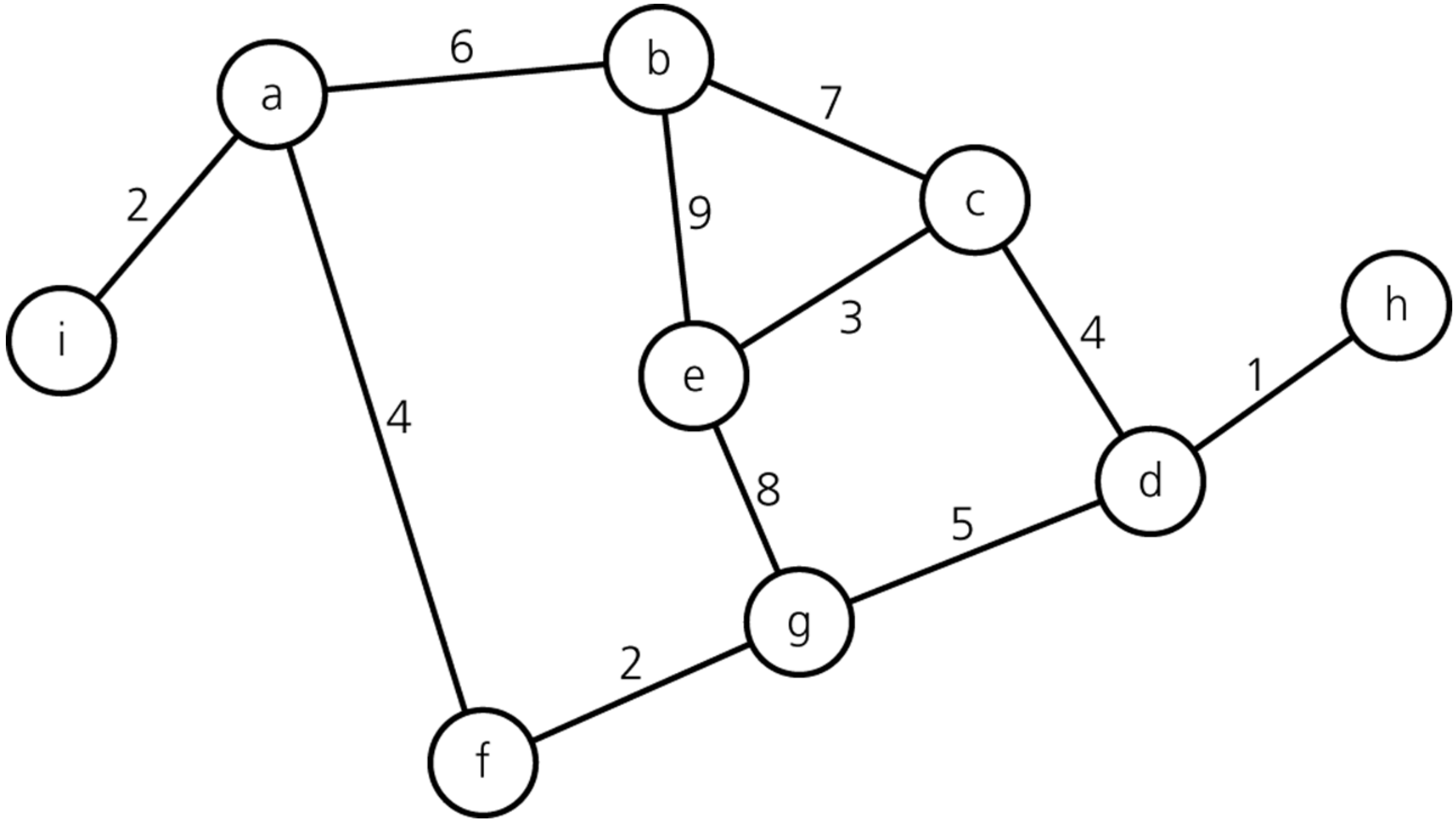
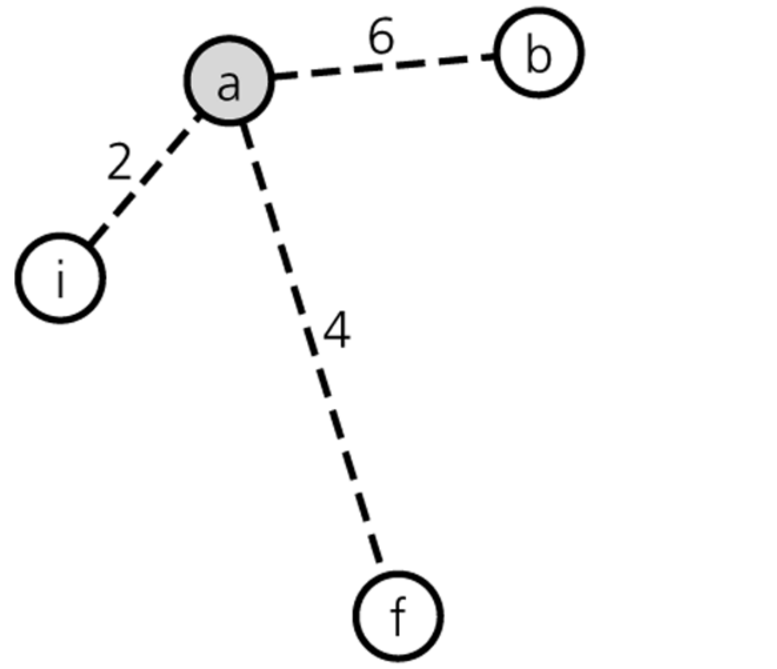
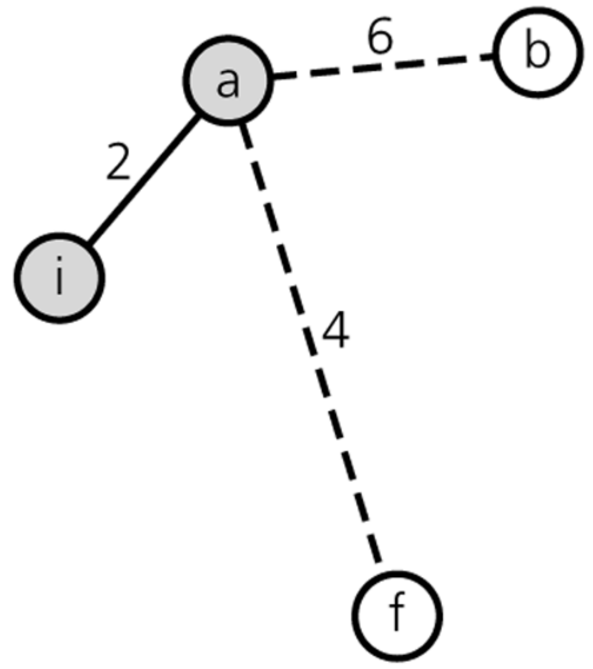


Figure 13.23a and 13.23b

A trace of ***PrimsAlgorithm*** for the graph in Figure 13-22, beginning a vertex *a*



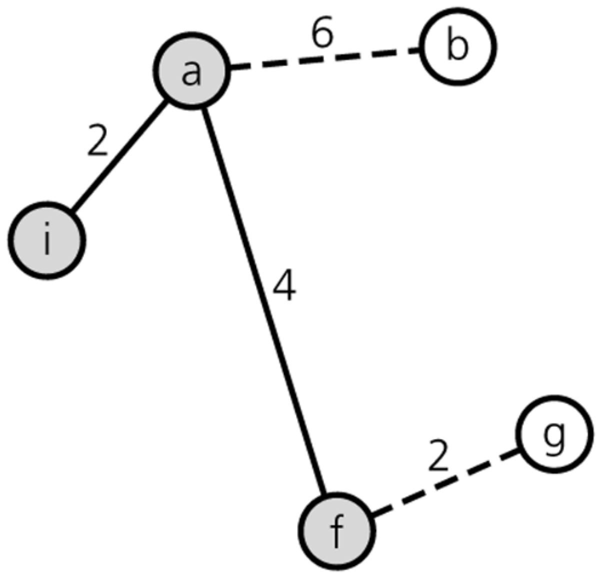
(a) Mark a, consider edges from a



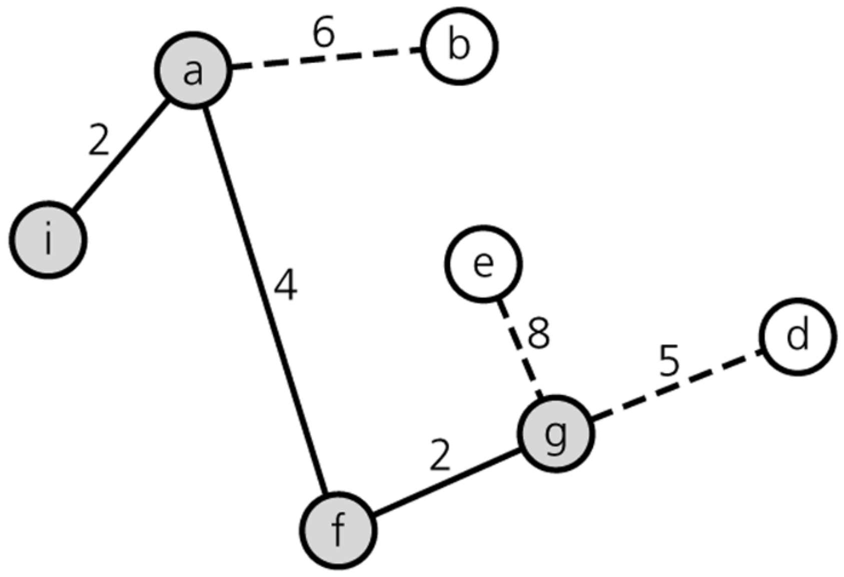
(b) Mark i, include edge (a, i)

Figure 13.23c and 13.23d

A trace of ***`PrimsAlgorithm`*** for the graph in Figure 13-22, beginning a vertex *a*



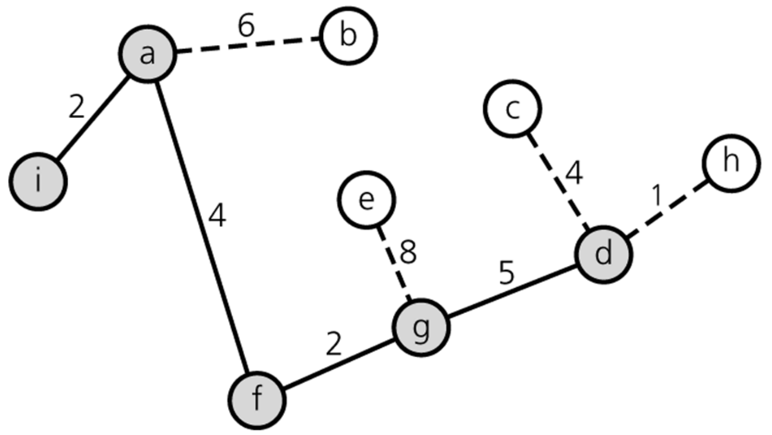
(c) Mark f, include edge (a, f)



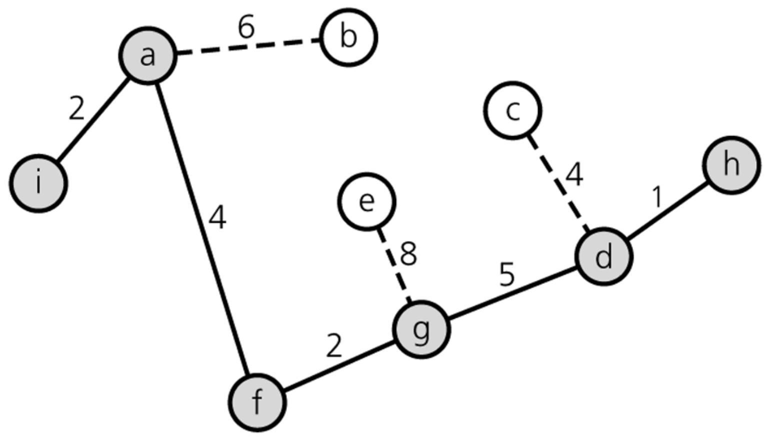
(d) Mark g, include edge (f, g)

Figure 13.23e and 13.23f

A trace of ***`PrimsAlgorithm`*** for the graph in Figure 13-22, beginning a vertex *a*



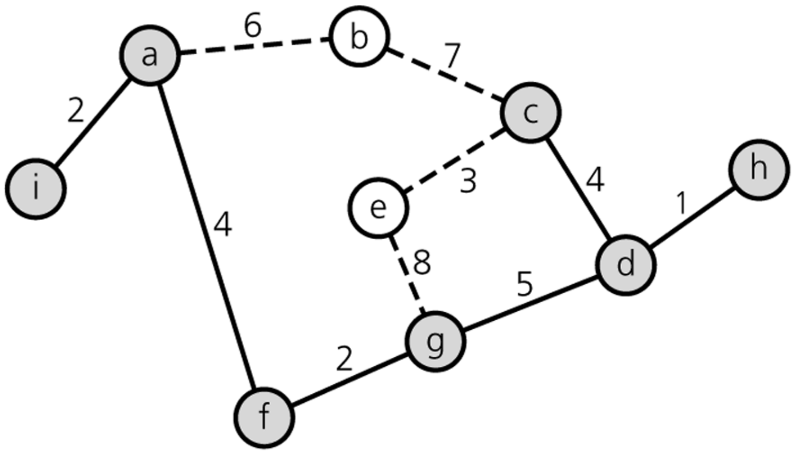
(e) Mark d, include edge (g, d)



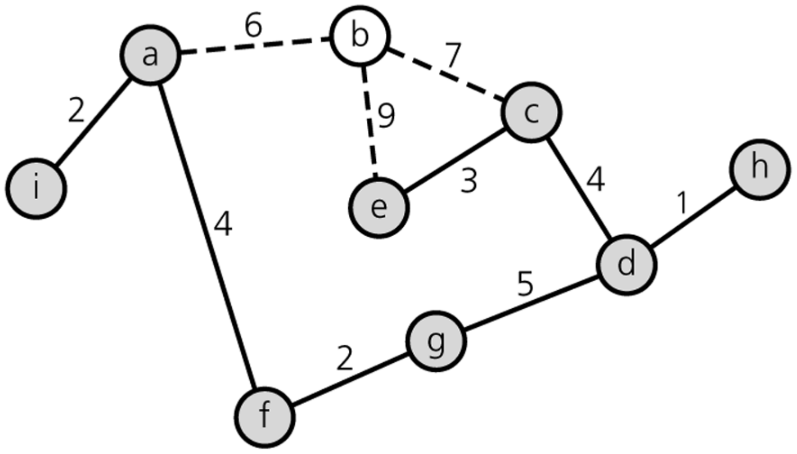
(f) Mark h, include edge (d, h)

Figure 13.23g and 13.23h

A trace of ***PrimsAlgorithm*** for the graph in Figure 13-22, beginning a vertex *a*



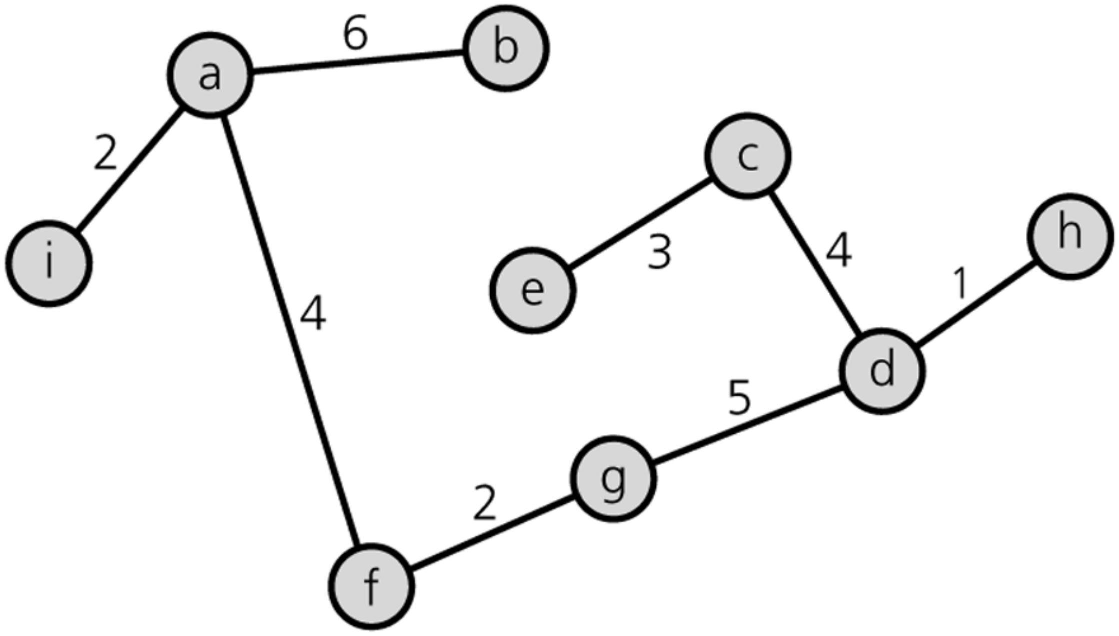
(g) Mark c, include edge (d, c)



(h) Mark e, include edge (c, e)

Figure 13.23i

A trace of ***PrimsAlgorithm*** for the graph in Figure 13-22, beginning a vertex *a*

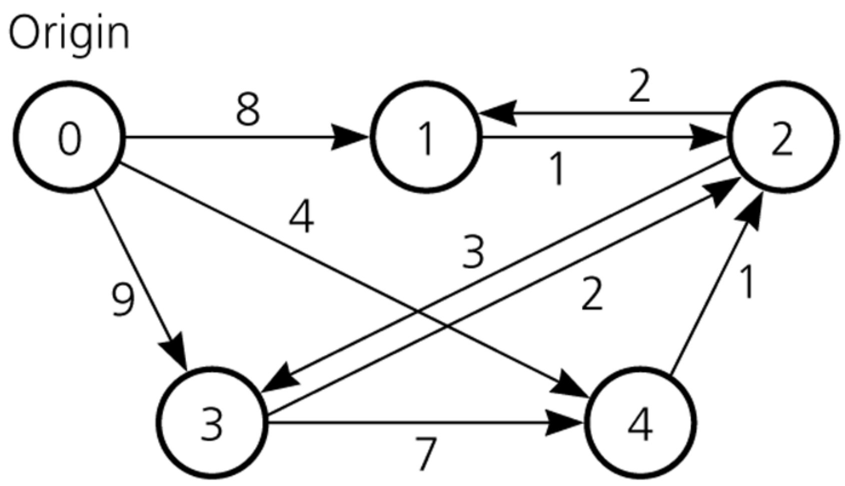


(i) Mark b, include edge (a, b)

Figure 13.24

a) A weighted directed graph and b) its adjacency matrix

(a)



(b)

$$\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & \infty & 8 & \infty & 9 & 4 \\
1 & \infty & \infty & 1 & \infty & \infty \\
2 & \infty & 2 & \infty & 3 & \infty \\
3 & \infty & \infty & 2 & \infty & 7 \\
4 & \infty & \infty & 1 & \infty & \infty
\end{array}$$

Figure 13.25

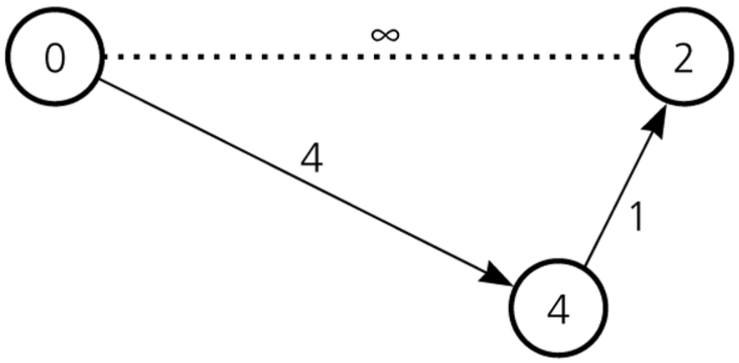
A trace of the shortest-path algorithm applied to the graph in Figure 13-24a

Step	v	vertexSet	weight[0]	weight[1]	weight[2]	weight[3]	weight[4]
1	–	0	0	8	∞	9	4
2	4	0, 4	0	8	5	9	4
3	2	0, 4, 2	0	7	5	8	4
4	1	0, 4, 2, 1	0	7	5	8	4
5	3	0, 4, 2, 1, 3	0	7	5	8	4

Figure 13.26a and b

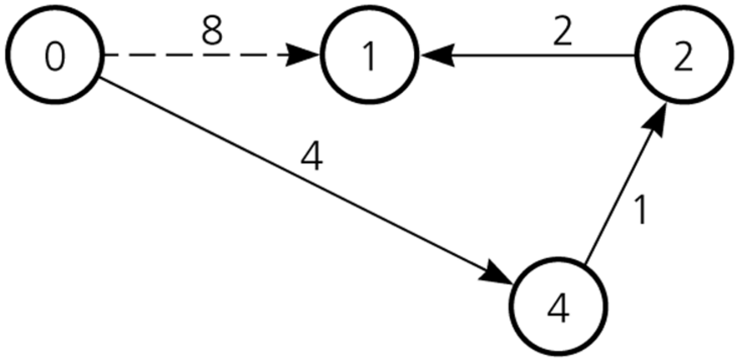
Checking **`weight[u]`** by examining the graph: a) **`weight[2]`** in Step 2; b) **`weight[1]`** in Step 3

(a)



Step 2. The path 0–4–2 is shorter than 0–2

(b)

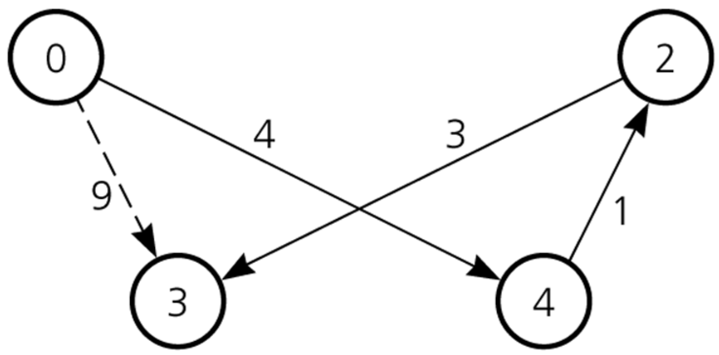


Step 3. The path 0–4–2–1 is shorter than 0–1

Figure 13.26c and d

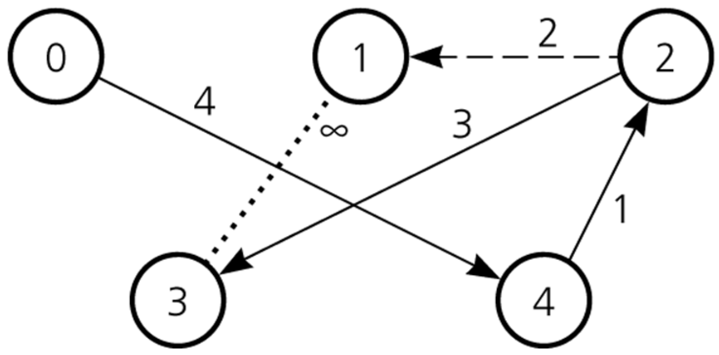
Checking ***weight*[u]** by examining the graph: c) ***weight*[3]** in Step 3; b) ***weight*[3]** in Step 4

(c)



Step 3 continued. The path 0–4–2–3 is shorter than 0–3

(d)



Step 4. The path 0–4–2–3 is shorter than 0–4–2–1–3

Figure 13.27

a) Euler's bridge problem and b) its multigraph representation

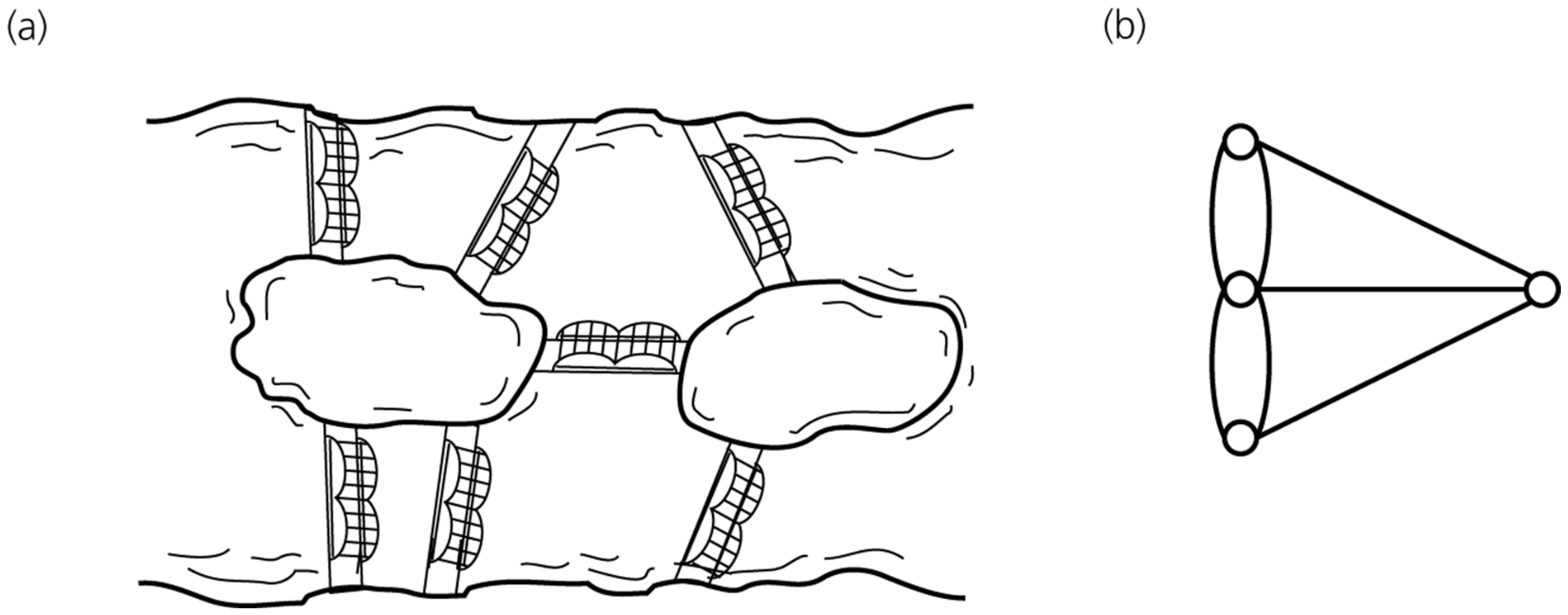


Figure 13.28

Pencil and paper drawings

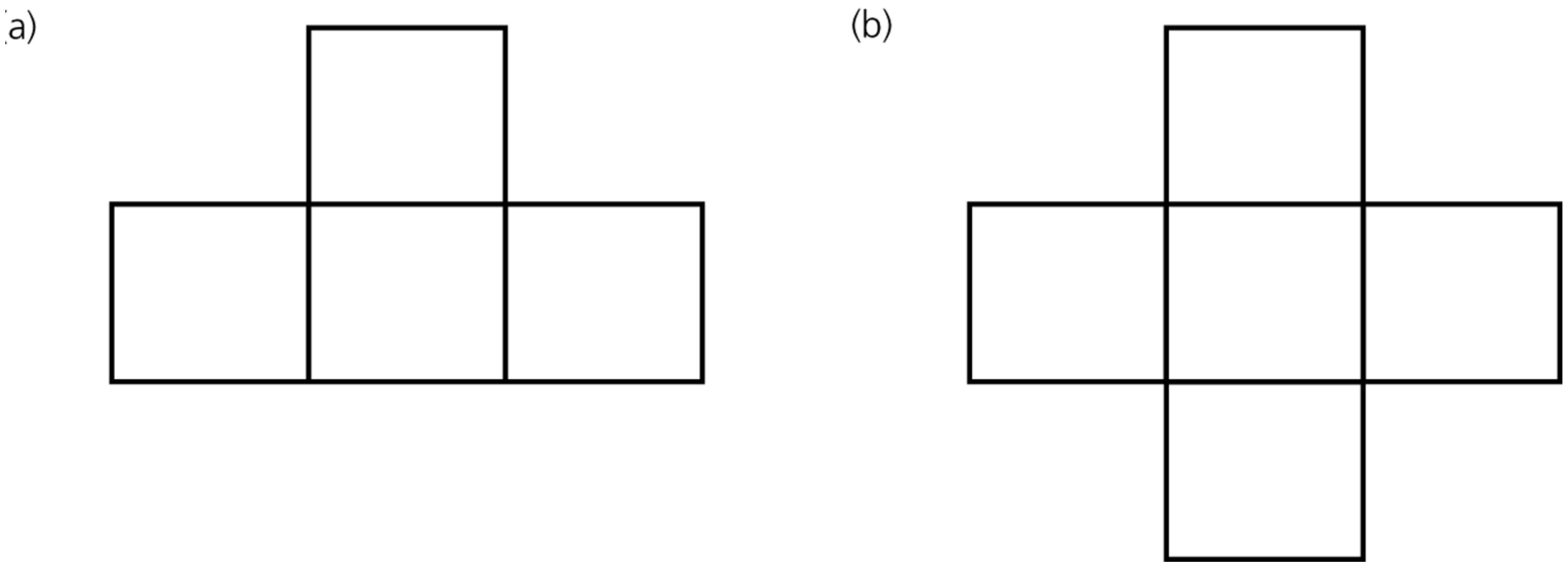
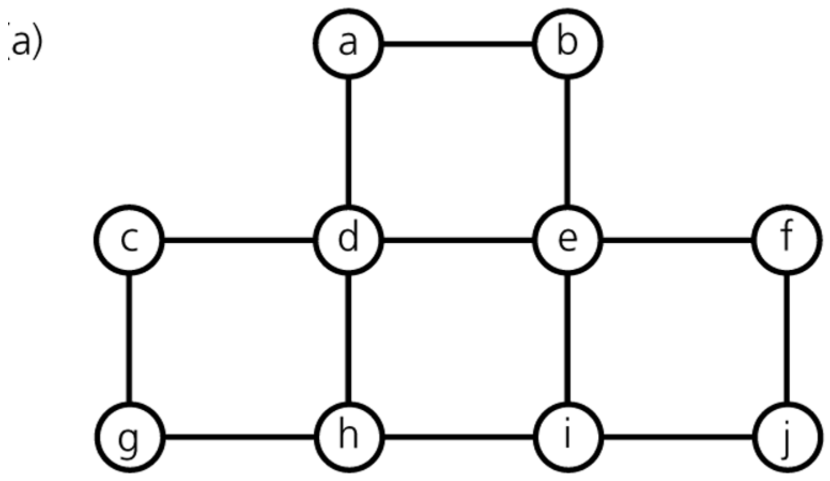


Figure 13.29

Connected undirected graphs based on the drawings in Figure 13-28



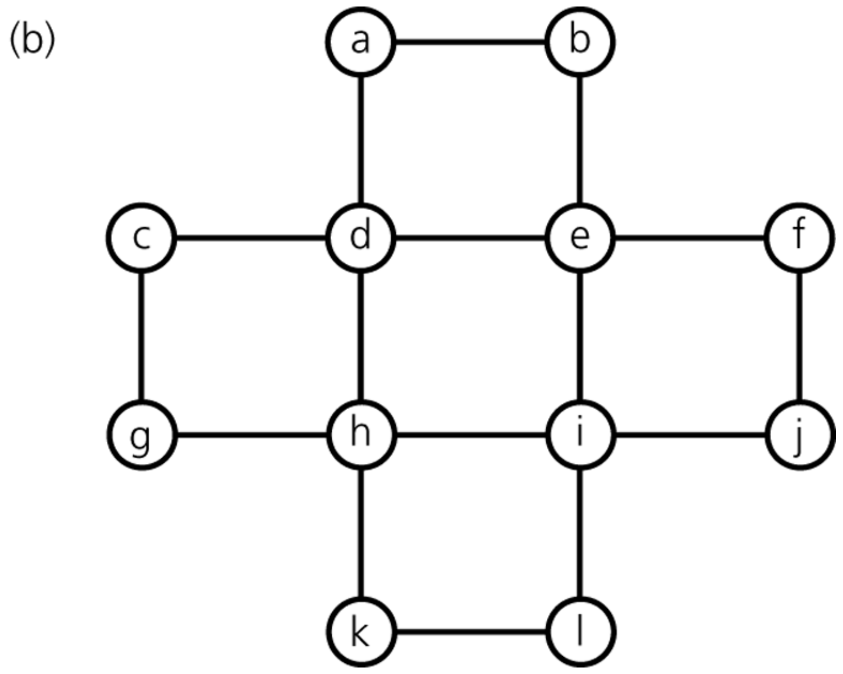
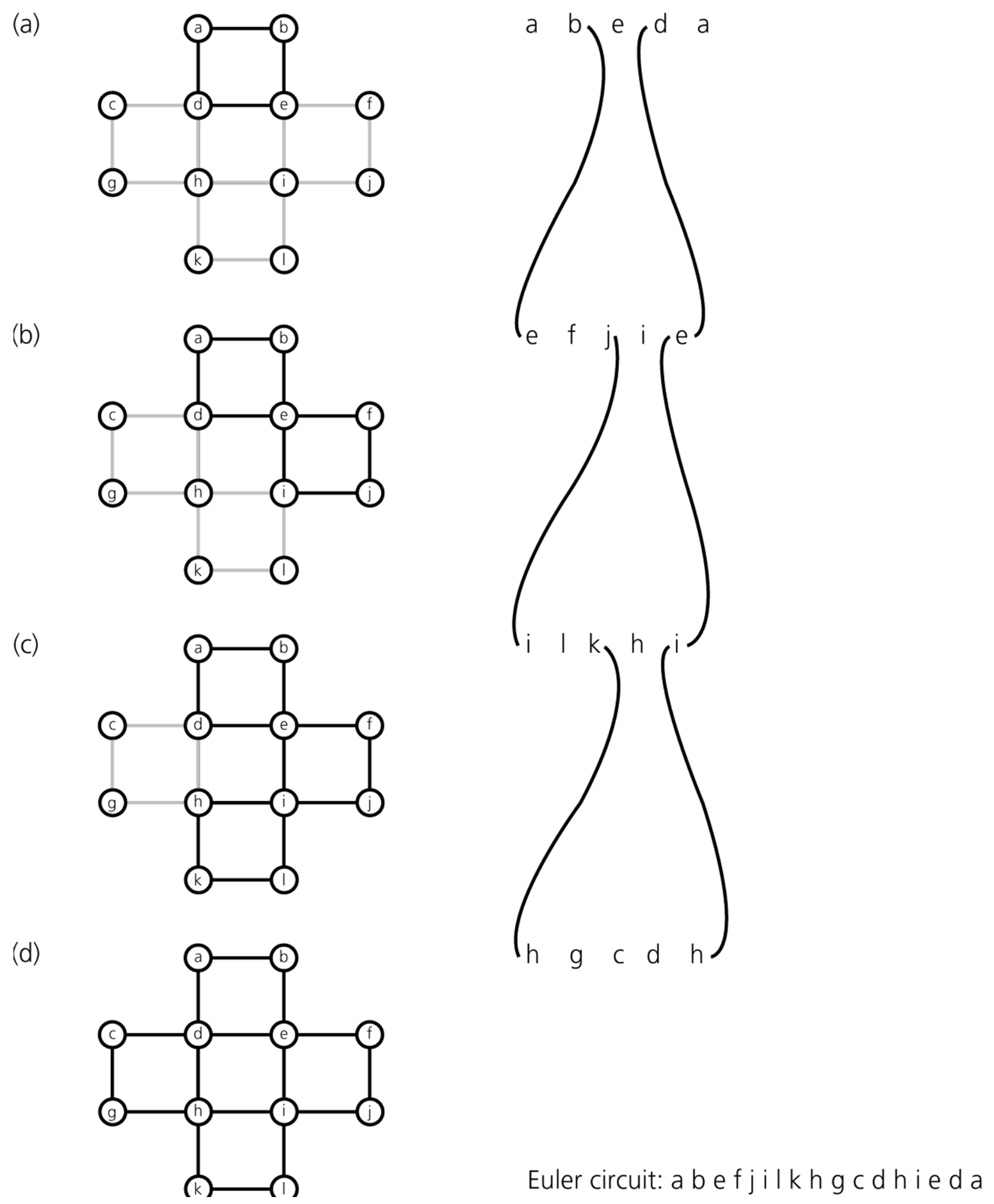


Figure 13.30

The steps to determine an Euler circuit for the graph in Figure 13-29b



Euler circuit: a b e f j i l k h g c d h i e d a

Figure 13.31

The three utilities problem

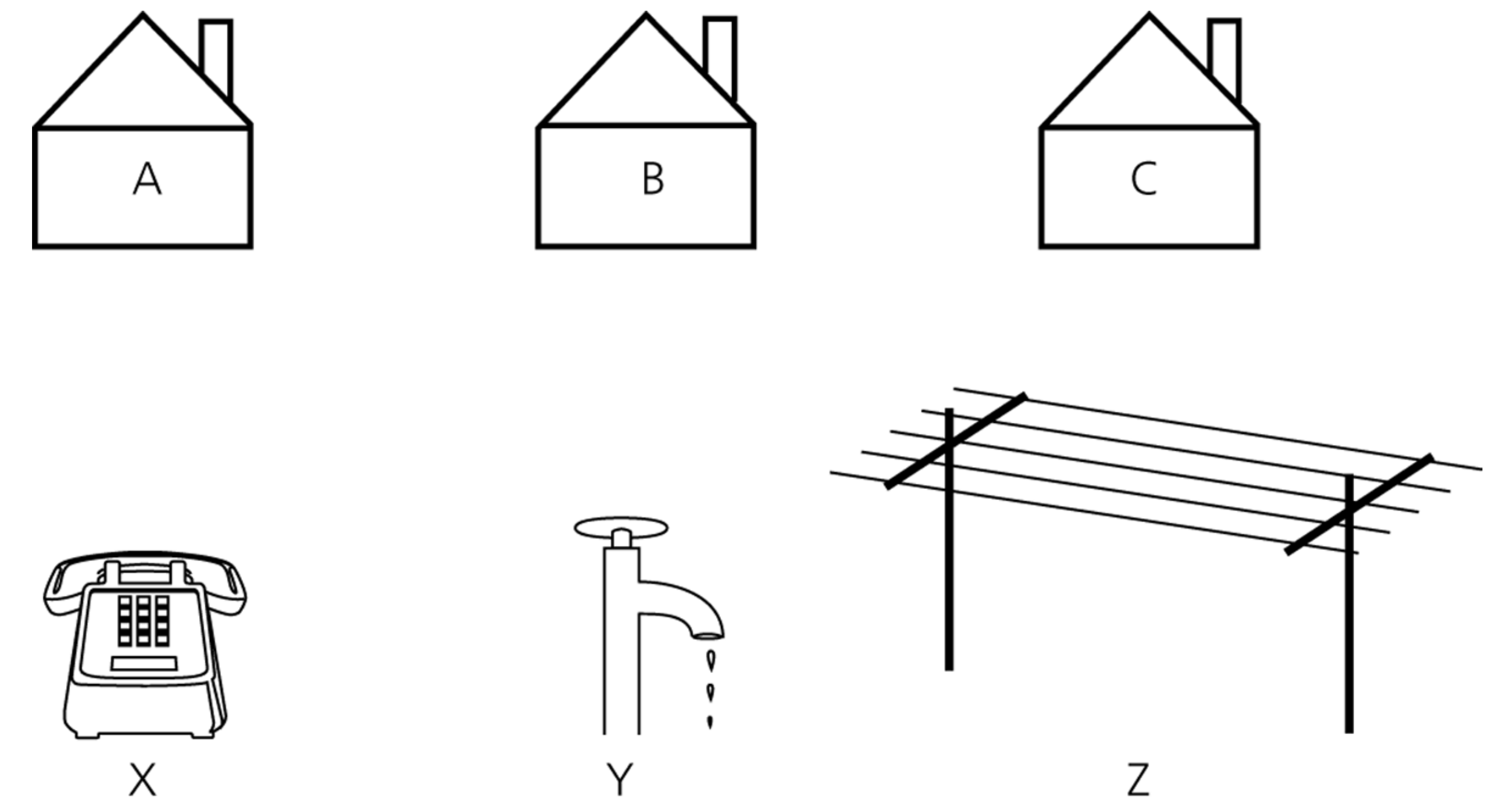


Figure 13.32

Graphs for Self-Test Exercises 1, 2, and 3

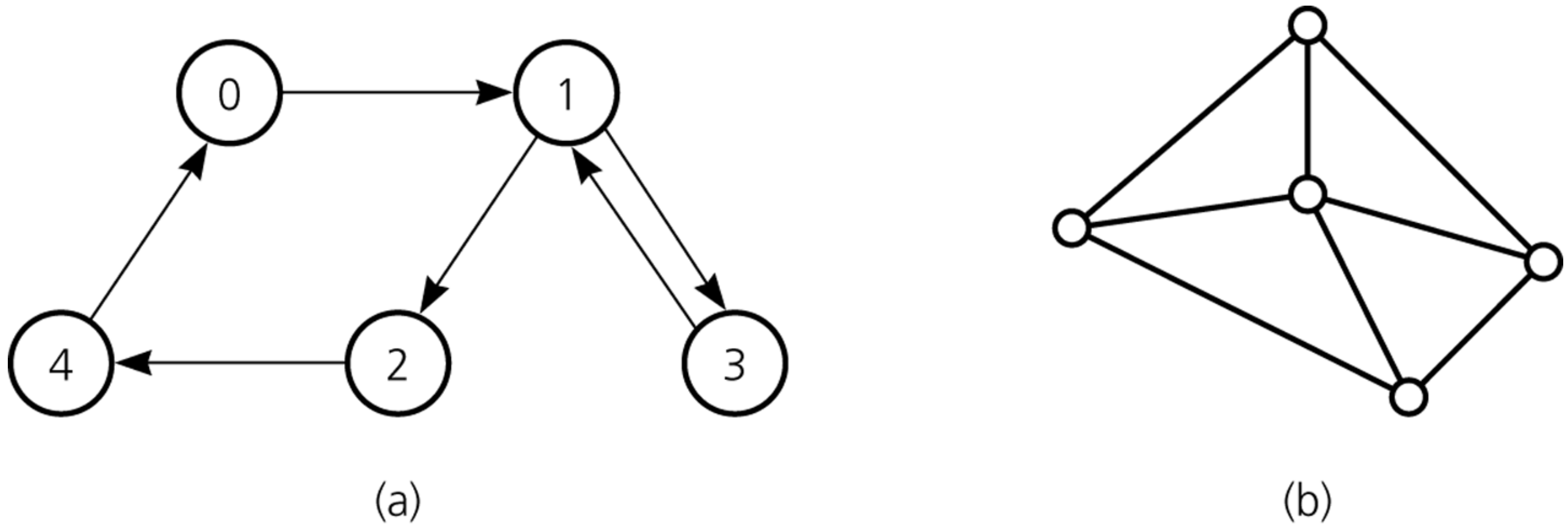


Figure 13.33

A graph for Self-Test Exercises 6 and 7 and Exercises 1 and 3

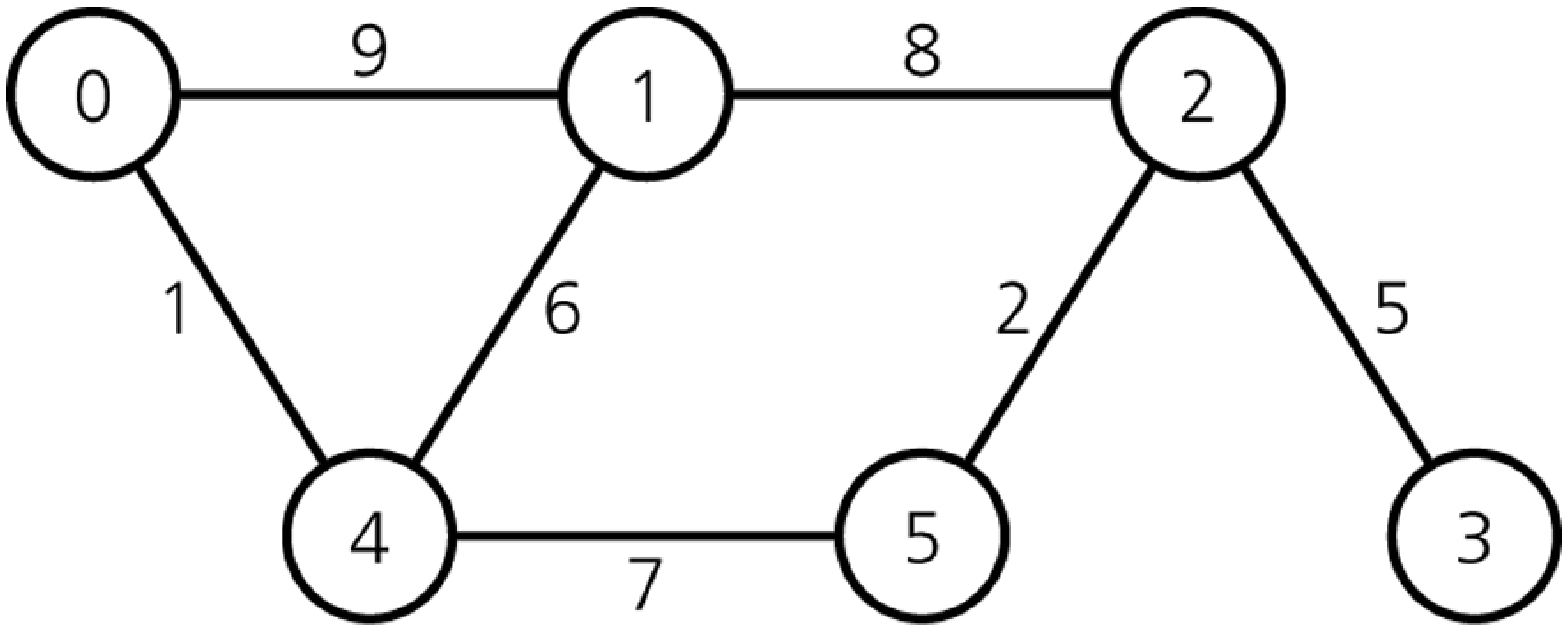


Figure 13.34

A graph for Exercise 1

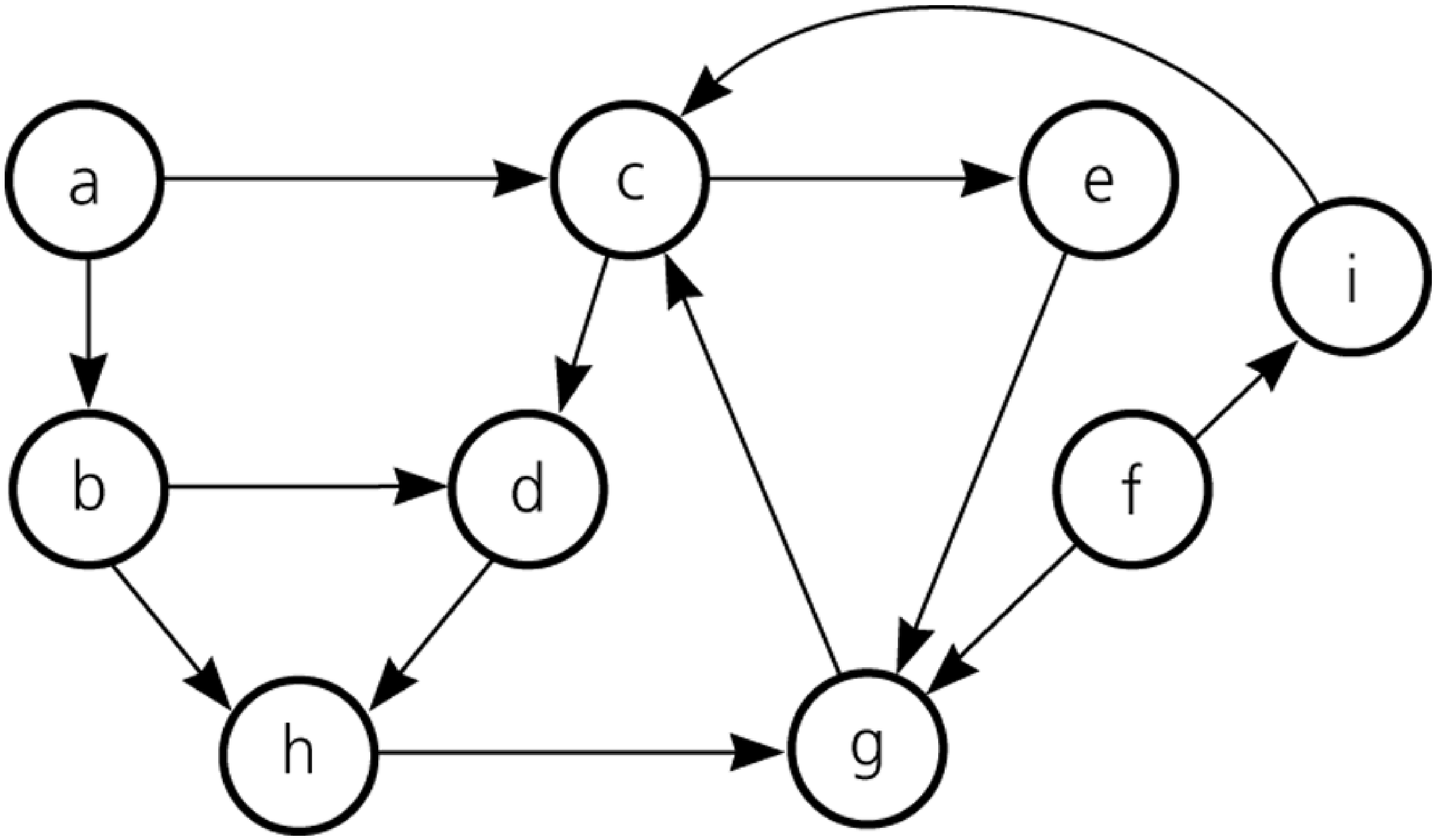


Figure 13.35

A graph for Exercises 3 and 8

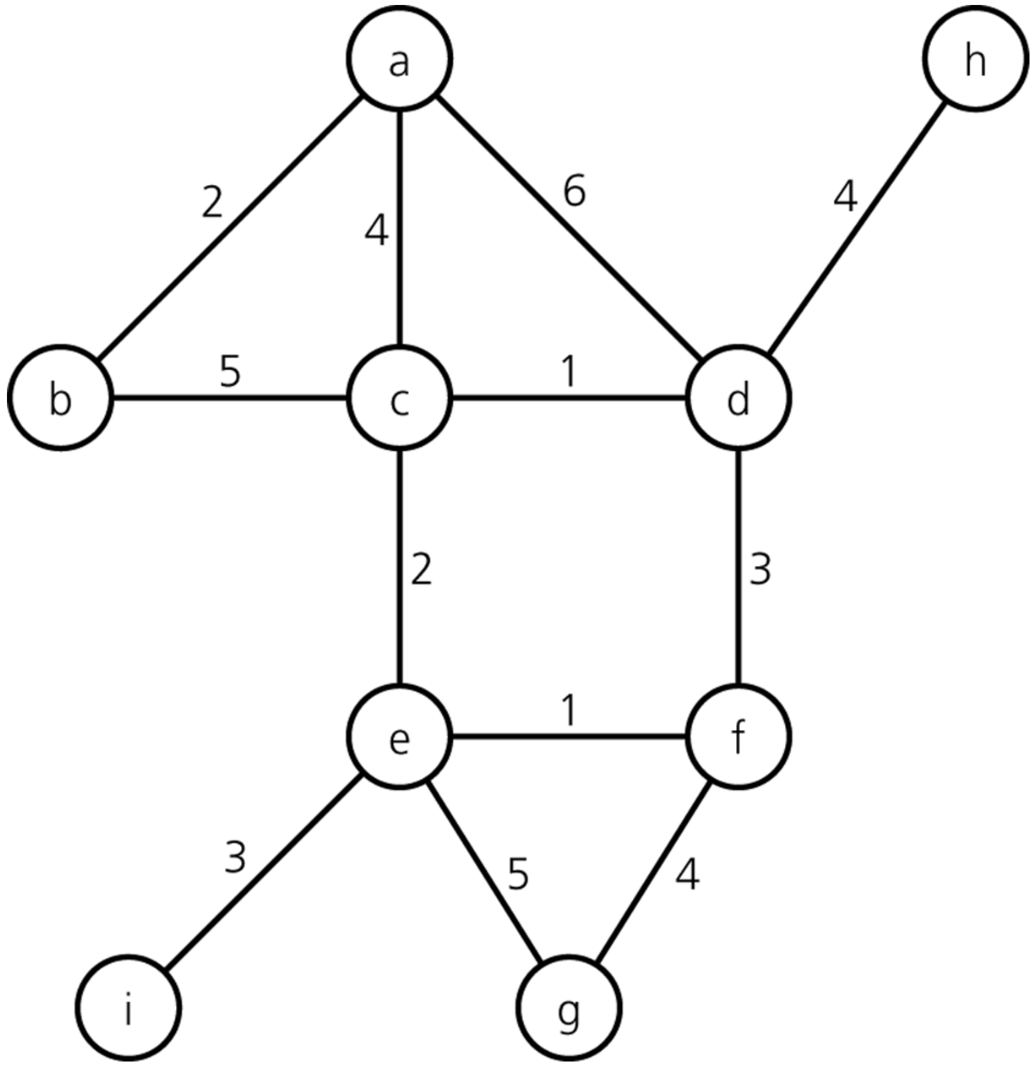


Figure 13.36

Graphs for Exercises 5 and 6

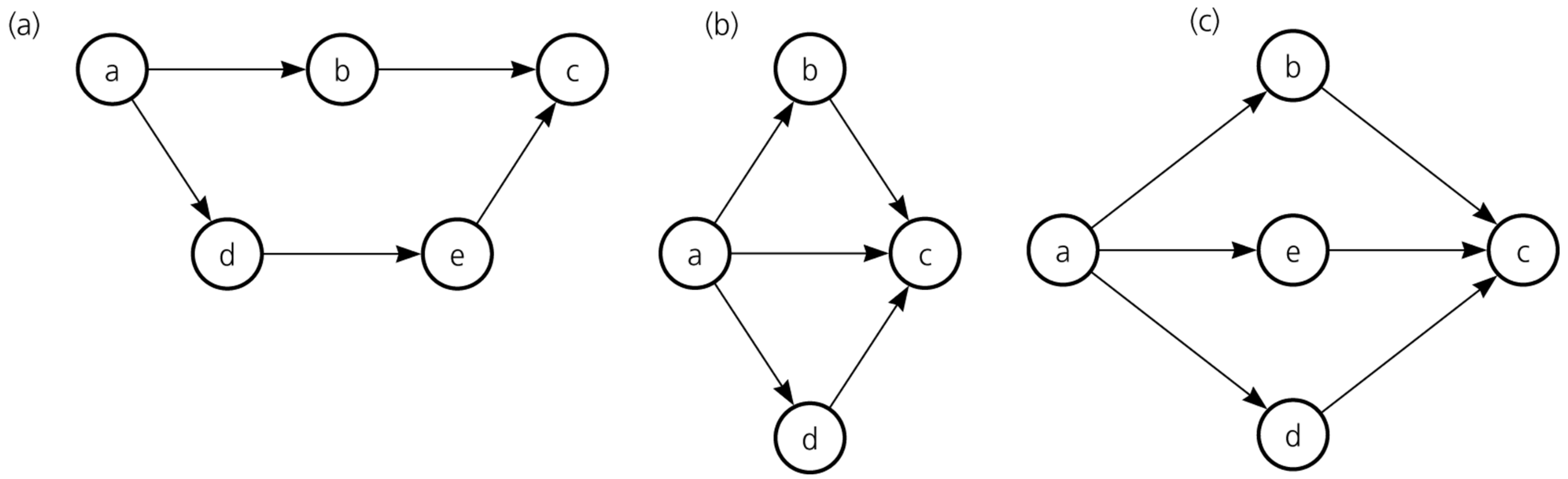


Figure 13.37

A graph for Exercise 11

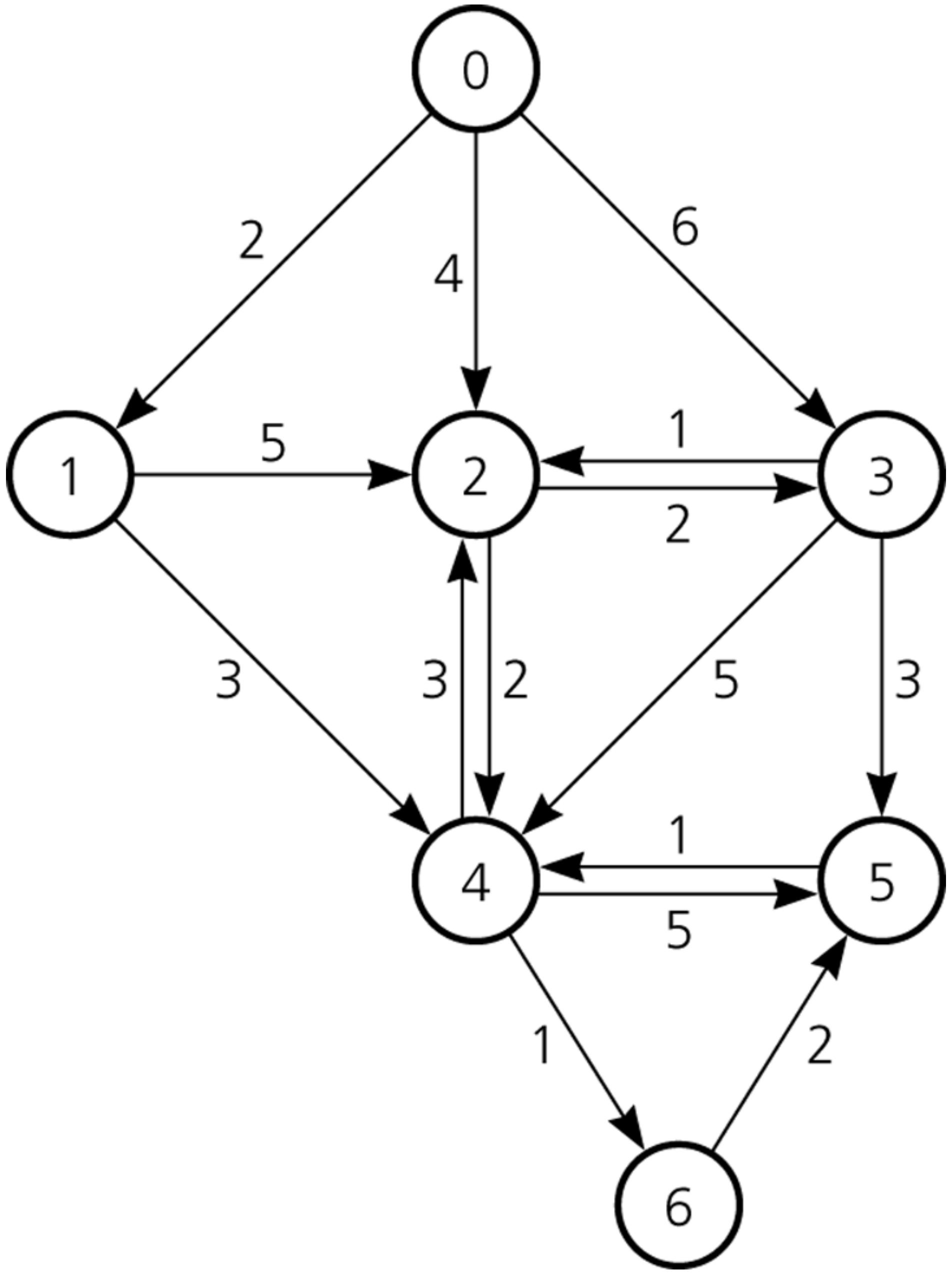


Figure 13.38

A graph for Exercise 13