CHAPTER 6: Graph Algorithms

This chapter contains (naturally) a number of different graph algorithms. Section 6.1 explores the relationship between directed graphs, associated binary relations, and adjacency matrices. It is seen that the earlier topic of the transitive closure of a binary relation relates to reachability. The two different algorithms for computing the reachability matrix - a "brute force" approach and Warshall's algorithm - provide a good illustration of an order of magnitude improvement obtained by use of a clever algorithm.

The Euler path algorithm (Section 6.2) is interesting because it answers an old children's game that the students have all seen. The contrast of the Euler path problem with the seemingly-similar Hamiltonian circuit problem is well worth mentioning, especially as a pointer to the brief discussion of computational complexity in Chapter 8. Shortest-path and minimal spanning tree algorithms (Section 6.3) introduce the idea of a greedy algorithm; additional shortest-path and minimal spanning tree algorithms appear in the exercises. These algorithms, as well as those for depth-first and breadth-first search (Section 6.4), are also subjected to (somewhat informal) analysis of algorithm techniques.

Students can readily appreciate the applicability of the shortest-path algorithm for routing in computer networks and the minimal spanning tree algorithm for network design Applications of depth-first search to reachability in a directed graph and to topological sorting are also discussed.

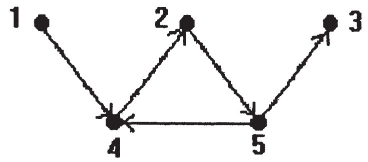
Despite these motivating applications, there may be good reason to pick and choose what you cover in this chapter, or even to skip it entirely. A data structures course will probably cover many of these algorithms.

EXERCISES 6.1

*1. $A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ $\rho = \{(1, 1), (2, 1), (2, 3), (3, 2)\}$

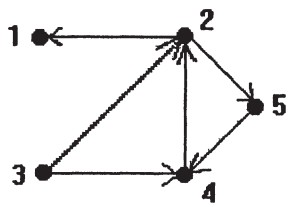
2. $A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $\rho = \{(1, 3), (2, 3), (3, 4), (4, 4)\}$

*3.



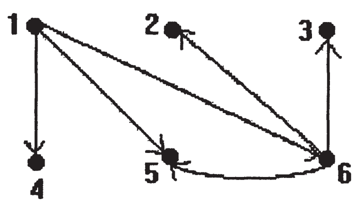
$\rho = \{(1, 4), (2, 5), (4, 2), (5, 3), (5, 4)\}$

4.



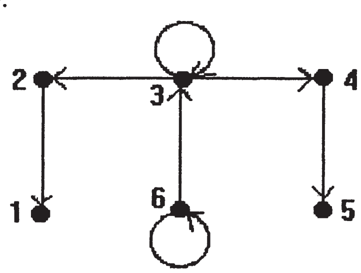
$\rho = \{(2, 1), (2, 5), (3, 2), (3, 4), (4, 2), (5, 4)\}$

5.



$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

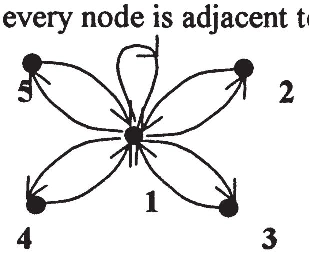
6.



$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

7. For every pair of nodes a and b, if there is an arc from a to b, then there is also an arc from b to a.

*8. The graph is a "star" with node 1 at the center, i.e., 1 is adjacent to every node and every node is adjacent to 1, but no other nodes are adjacent.



9. The graph is a cycle through all the nodes.

10. If entry i,j = 1, then entry j,i = 0 unless i = j.

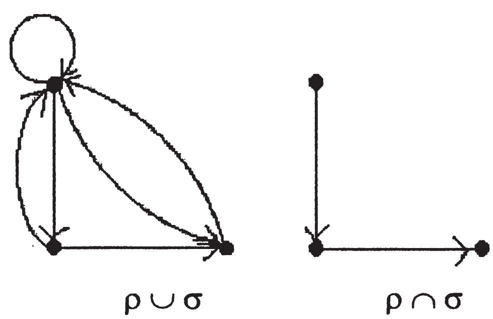
11. The adjacency matrix for $\rho \cup \sigma$ is

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

The adjacency matrix for $\rho \cap \sigma$ is

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

*12.



13. $A^2 = \begin{bmatrix} 2 & 1 & 2 & 1 \\ 1 & 0 & 0 & 1 \\ 2 & 1 & 1 & 1 \\ 2 & 1 & 0 & 2 \end{bmatrix}$ $\qquad A^{(2)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$

14. Because there is a path from every node to every other node, the matrix R will have all 1 entries.

*15. $R = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ $\qquad$ 16. $R = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

17. $A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ $A^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ $A^{(3)} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

$$R = A \vee A^{(2)} \vee A^{(3)} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

18. $R = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

*19. $R = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$

20. $R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}$

21. $R = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$

22. $R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

*23. $A = M_0 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ $M_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ $M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ $M_3 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = R$

24. $R = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

25. $R = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$

26. $R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}$

*27. $R = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$

28. $R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

*29. $A^2[i,j] = \sum_{k=1}^{n} a_{ik}a_{kj}$

If a term such as $a_{i2}a_{2j}$ in this sum is 0, then either $a_{i2} = 0$ or $a_{2j} = 0$ (or both) and there is either no path of length 1 from n_i to n_2 or no path of length 1 from n_2 to n_j (or both). Thus there are no paths of length 2 from n_i to n_j passing through n_2. If $a_{i2}a_{2j} \neq 0$, then $a_{i2} = p$ and $a_{2j} = q$, where p and q are positive integers. Then there are p paths of length 1 from n_i to n_2 and q paths of length 1 from n_2 to n_j. By the Multiplication Principle, there are pq possible paths of length 2 from n_i to n_j through n_2. By the Addition Principle, the sum of all such terms gives all possible paths of length 2 from n_i to n_j.

30. The proof is by induction on n. The result for n = 1 is true by the definition of A. (The result for n = 2 is true by Exercise 29 above.) Assume that $A^p[i,j]$ equals the number of paths of length p from n_i to n_j. Then

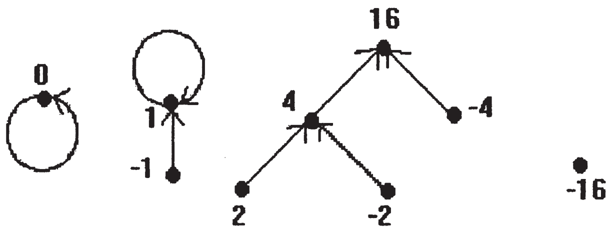
$$A^{p+1}[i,j] = \sum_{k=1}^{n} A^p[i,k]a_{kj}$$

A term such as $A^p[i,q]a_{qj}$ is the product of the number of paths of length p from n_i to n_q and the number of paths of length 1 from n_q to n_j, which is the number of paths of length p + 1 from n_i to n_j through n_q. The sum of all such terms gives all possible paths of length p + 1 from n_i to n_j.

31. 3; $A^2 = \begin{bmatrix} 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

32. 6; $A^4 = \begin{bmatrix} 1 & 0 & 4 & 2 & 6 \\ 0 & 1 & 4 & 2 & 6 \\ 0 & 0 & 3 & 2 & 6 \\ 0 & 0 & 2 & 1 & 4 \\ 0 & 0 & 4 & 2 & 7 \end{bmatrix}$

33.



EXERCISES 6.2

1. The graph of Example 3, Chapter 2, has 4 nodes of degree 3, hence, by the theorem on Euler paths, it has no Euler path.

*2. yes 3. yes 4. no 5. no

6. yes *7. no 8. yes 9. no 10. yes

*11. $\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$

total after row 2 is 0

12. $\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$

total after row 4 is 2

*13. $\begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$

$i = 8$

14. $\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$

$i = 5$

*15. no 16. yes 17. yes 18. no

19. yes *20. yes 21. yes 22. no

23.

A path through the graph exists that uses each arc exactly once; there is no requirement that each node be visited.

*24. Any two nodes must be part of the Hamiltonian circuit, therefore there is a path between them, namely that part of the circuit that is between them.

25. a. $(n - 1)^n$ (At each of the n nodes that begins an arc of the path, there are n - 1 possible arcs to use.)

b. $(n - 1)(n - 2)^{n-1}$ (There are n - 1 choices for the first arc of the path, but only n - 2 choices for each arc after that because the same arc cannot be repeated.)

c. $(n - 1)!$ (There are n - 1 choices for the first arc, n - 2 for the second, etc.)

26. No; using rooms as nodes (plus an outside node), and doorways as arcs, the resulting graph has 4 odd vertices, hence no Euler path.

*27. a. $n = 2$ or $n =$ any odd number
b. $n > 2$

28. a. Euler paths exist for $m = n = 1$, for $m = 2$ and $n =$ any odd number (or vice versa), or for m even and n even.
b. Hamiltonian circuits exist for $m = n \geq 2$

29. Each odd vertex is the beginning or end of such a path, so there exist at least n such paths. If each pair of odd vertices is connected by a temporary arc, the resulting graph has no odd vertices and has an Euler cycle. The removal of the temporary arcs leaves $\leq n$ disjoint Euler paths which traverse the original graph. Therefore n paths are necessary and sufficient.

*30. Begin at any node and take one of the arcs out from that node. Each time a new node is entered on an arc, there is exactly one unused arc on which to exit that node; because the arc is unused, it will lead to a new node or to the initial node. Upon return to the initial node, if all nodes have been used, we are done. If there is an unused node, because the graph is connected, there is an unused path from that node to a used node, which means the used node has degree ≥ 3, a contradiction.

EXERCISES 6.3

*1. IN = {2}

	1	2	3	4	5	6	7	8
d	3	0	2	∞	∞	∞	1	∞
s	2	-	2	2	2	2	2	2

p = 7, IN = {2,7}

	1	2	3	4	5	6	7	8
d	3	0	2	∞	∞	6	1	2
s	2	-	2	2	2	7	2	7

p = 3, IN = {2,7,3}

	1	2	3	4	5	6	7	8
d	3	0	2	3	∞	6	1	2
s	2	-	2	3	2	7	2	7

p = 8, IN = {2,7,3,8}

	1	2	3	4	5	6	7	8
d	3	0	2	3	3	6	1	2
s	2	-	2	3	8	7	2	7

p = 5, IN = {2,7,3,8,5}

	1	2	3	4	5	6	7	8
d	3	0	2	3	3	6	1	2
s	2	-	2	3	8	7	2	7

path: 2,7,8,5 distance = 3

2. IN = {3}

	1	2	3	4	5	6	7	8
d	5	2	0	1	∞	∞	∞	2
s	3	3	-	3	3	3	3	3

p = 4, IN = {3,4}

	1	2	3	4	5	6	7	8
d	5	2	0	1	5	∞	∞	2
s	3	3	-	3	4	3	3	3

p = 2, IN = {3,4,2}

	1	2	3	4	5	6	7	8
d	5	2	0	1	5	∞	4	2
s	3	3	-	3	4	3	2	3

p = 8, IN = {3,4,2,8}

	1	2	3	4	5	6	7	8
d	5	2	0	1	3	∞	3	2
s	3	3	-	3	8	3	8	3

p = 5, IN = {3,4,2,8,5}

	1	2	3	4	5	6	7	8
d	5	2	0	1	3	9	3	2
s	3	3	-	3	8	5	8	3

p = 7, IN = {3,4,2,8,5,7}

	1	2	3	4	5	6	7	8
d	5	2	0	1	3	8	3	2
s	3	3	-	3	8	7	8	3

p = 1, IN = {3,4,2,8,5,7,1}

	1	2	3	4	5	6	7	8
d	5	2	0	1	3	6	3	2
s	3	3	-	3	8	1	8	3

p = 6, IN = {3,4,2,8,5,7,1,6}

	1	2	3	4	5	6	7	8
d	5	2	0	1	3	6	3	2
s	3	3	-	3	8	1	8	3

path: 3,1,6 (alternate
path: 3,2,1,6) distance = 6

3. IN = {1}

	1	2	3	4	5	6	7	8
d	0	3	5	∞	8	1	∞	∞
s	-	1	1	1	1	1	1	1

p = 6, IN = {1,6}

	1	2	3	4	5	6	7	8
d	0	3	5	∞	7	1	6	∞
s	-	1	1	1	6	1	6	1

p = 2, IN = {1,6,2}

	1	2	3	4	5	6	7	8
d	0	3	5	∞	7	1	4	∞
s	-	1	1	1	6	1	2	1

p = 7, IN = {1,6,2,7}

	1	2	3	4	5	6	7	8
d	0	3	5	∞	7	1	4	5
s	-	1	1	1	6	1	2	7

p = 3, IN = {1,6,2,7,3}

	1	2	3	4	5	6	7	8
d	0	3	5	6	7	1	4	5
s	-	1	1	3	6	1	2	7

p = 8, IN = {1,6,2,7,3,8}

	1	2	3	4	5	6	7	8
d	0	3	5	6	6	1	4	5
s	-	1	1	3	8	1	2	7

p =5, IN = {1,6,2,7,3,8,5}

	1	2	3	4	5	6	7	8
d	0	3	5	6	6	1	4	5
s	-	1	1	3	8	1	2	7

path: 1,2,7,8,5 distance = 6

4.

IN = {4}

	1	2	3	4	5	6	7	8
d	∞	∞	1	0	4	∞	∞	∞
s	4	4	4	-	4	4	4	4

p = 3, IN = {4,3}

	1	2	3	4	5	6	7	8
d	6	3	1	0	4	∞	∞	3
s	3	3	4	-	4	4	4	3

p = 2, IN = {4,3,2}

	1	2	3	4	5	6	7	8
d	6	3	1	0	4	∞	4	3
s	3	3	4	-	4	4	2	3

p = 8, IN = {4,3,2,8}

	1	2	3	4	5	6	7	8
d	6	3	1	0	4	∞	4	3
s	3	3	4	-	4	4	2	3

p = 7, IN = {4,3,2,8,7}

	1	2	3	4	5	6	7	8
d	6	3	1	0	4	9	4	3
s	3	3	4	-	4	7	2	3

path: 4,3,2,7 distance = 4

*5.

IN = {a}

	a	b	c	d	e	f
d	0	1	3	∞	∞	∞
s	-	a	a	a	a	a

p = b, IN = {a,b}

	a	b	c	d	e	f
d	0	1	2	∞	∞	2
s	-	a	b	a	a	b

p = c, IN = {a,b,c}

	a	b	c	d	e	f
d	0	1	2	4	6	2
s	-	a	b	c	c	b

p = f, IN = {a,b,c,f}

	a	b	c	d	e	f
d	0	1	2	4	3	2
s	-	a	b	c	f	b

p = e, IN = {a,b,c,f,e}

	a	b	c	d	e	f
d	0	1	2	4	3	2
s	-	a	b	c	f	b

path: a,b,f,e distance = 3

6.

IN = {d}

	a	b	c	d	e	f
d	∞	∞	2	0	1	2
s	d	d	d	-	d	d

p = e, IN = {d,e}

	a	b	c	d	e	f
d	∞	∞	2	0	1	2
s	d	d	d	-	d	d

p = c, IN = {d,e,c}

	a	b	c	d	e	f
d	5	3	2	0	1	2
s	c	c	d	-	d	d

p = f, IN = {d,e,c,f}

	a	b	c	d	e	f
d	5	3	2	0	1	2
s	c	c	d	-	d	d

p = b, IN = {d,e,c,f,b}

	a	b	c	d	e	f
d	4	3	2	0	1	2
s	b	c	d	-	d	d

p = a, IN = {d,e,c,f,b,a}

	a	b	c	d	e	f
d	4	3	2	0	1	2
s	b	c	d	-	d	d

path: d,c,b,a distance = 4

*7.

IN = {1}

	1	2	3	4	5	6	7
d	0	2	∞	∞	3	2	∞
s	-	1	1	1	1	1	1

p = 2, IN = {1,2}

	1	2	3	4	5	6	7
d	0	2	3	∞	3	2	∞
s	-	1	2	1	1	1	1

p = 6, IN = {1,2,6}

	1	2	3	4	5	6	7
d	0	2	3	∞	3	2	5
s	-	1	2	1	1	1	6

p = 3, IN = {1,2,6,3}

	1	2	3	4	5	6	7
d	0	2	3	4	3	2	5
s	-	1	2	3	1	1	6

p = 5, IN = {1,2,6,3,5}

	1	2	3	4	5	6	7
d	0	2	3	4	3	2	5
s	-	1	2	3	1	1	6

p = 4, IN = {1,2,6,3,5,4}

	1	2	3	4	5	6	7
d	0	2	3	4	3	2	5
s	-	1	2	3	1	1	6

p = 7, IN = {1,2,6,3,5,4,7}

	1	2	3	4	5	6	7
d	0	2	3	4	3	2	5
s	-	1	2	3	1	1	6

path: 1,6,7 distance = 5

8.

IN = {3}

	1	2	3	4	5	6	7
d	∞	∞	0	1	∞	∞	∞
s	3	3	-	3	3	3	3

p = 4, IN = {3,4}

	1	2	3	4	5	6	7
d	∞	∞	0	1	2	∞	2
s	3	3	-	3	4	3	4

p = 5, IN = {3,4,5}

	1	2	3	4	5	6	7
d	∞	∞	0	1	2	3	2
s	3	3	-	3	4	5	4

p = 7, IN = {3,4,5,7}

	1	2	3	4	5	6	7
d	∞	∞	0	1	2	3	2
s	3	3	-	3	4	5	4

p = 6, IN = {3,4,5,7,6}

	1	2	3	4	5	6	7
d	∞	∞	0	1	2	3	2
s	3	3	-	3	4	5	4

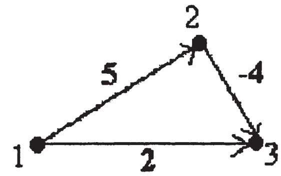
p = 1, IN = {3,4,5,7,6,1}

	1	2	3	4	5	6	7
d	∞	∞	0	1	2	3	2
s	3	3	-	3	4	5	4

No path from 3 to 1

9. a. Change the condition on the **while** loop to continue until all nodes are in IN. Also, rather than writing out a particular shortest path, make *d* and *s* output parameters that carry the information about shortest paths and their distances.
   b. No, the worst case was already computed assuming the destination node is the last node brought into IN.

10.



To find the shortest path from 1 to 3, the algorithm proceeds as follows:

I = {1}

	1	2	3
d	0	5	2
s	-	1	1

p = 3, IN = {1,3}

	1	2	3
d	0	5	2
s	-	1	1

Thus the algorithm will select the path 1-3 with distance 2, although the shortest path is 1-2-3 with distance 1. Allowing negative weights makes the greedy property insufficient for success, because a path with an initially high weight can later have its weight reduced by negative values, but this cannot be seen locally.

*11.

	1	2	3	4	5	6	7	8
d	3	0	2	∞	∞	∞	1	∞
s	2	-	2	2	2	2	2	2

(1)

	1	2	3	4	5	6	7	8
d	3	0	2	3	11	4	1	2
s	2	-	2	3	1	1	2	7

(2)

	1	2	3	4	5	6	7	8
d	3	0	2	3	3	4	1	2
s	2	-	2	3	8	1	2	7

(3)

No further changes in *d* or *s*.

12.

	1	2	3	4	5	6	7	8
d	0	3	5	∞	8	1	∞	∞
s	-	1	1	1	1	1	1	1

(1)

	1	2	3	4	5	6	7	8
d	0	3	5	6	7	1	4	7
s	-	1	1	3	6	1	2	3

(2)

	1	2	3	4	5	6	7	8
d	0	3	5	6	7	1	4	5
s	-	1	1	3	6	1	2	7

(3)

	1	2	3	4	5	6	7	8
d	0	3	5	6	6	1	4	5
s	-	1	1	3	8	1	2	7

(4)

No further changes in *d* or *s*.

13.

	1	2	3	4	5	6	7
d	0	2	∞	∞	3	2	∞
s	-	1	1	1	1	1	1

(1)

	1	2	3	4	5	6	7
d	0	2	3	∞	3	2	5
s	-	1	2	1	1	1	6

(2)

	1	2	3	4	5	6	7
d	0	2	3	4	3	2	5
s	-	1	2	3	1	1	6

(3)

No further changes in *d* or *s*.

14.

	1	2	3
d	0	5	2
s	-	1	1

(1)

	1	2	3
d	0	5	1
s	-	1	2

(2)

*15. Initial A and after k = x:

	x	1	2	3	y
x	0	1	∞	4	∞
1	1	0	3	1	5
2	∞	3	0	2	2
3	4	1	2	0	3
y	∞	5	2	3	0

after k = 1 and k = 2:

	x	1	2	3	y
x	0	1	4	2	6
1	1	0	3	1	5
2	4	3	0	2	2
3	2	1	2	0	3
y	6	5	2	3	0

after k = 3 and k = y:

	x	1	2	3	y
x	0	1	4	2	5
1	1	0	3	1	4
2	4	3	0	2	2
3	2	1	2	0	3
y	5	4	2	3	0

16. Initial A:

	1	2	3	4	5	6	7	8
1	0	3	5	∞	8	1	∞	∞
2	3	0	2	∞	∞	∞	1	∞
3	5	2	0	1	∞	∞	∞	2
4	∞	∞	1	0	4	∞	∞	∞
5	8	∞	∞	4	0	6	∞	1
6	1	∞	∞	∞	6	0	5	∞
7	∞	1	∞	∞	∞	5	0	1
8	∞	∞	2	∞	1	∞	1	0

after k = 1:

	1	2	3	4	5	6	7	8
1	0	3	5	∞	8	1	∞	∞
2	3	0	2	∞	11	4	1	∞
3	5	2	0	1	13	6	∞	2
4	∞	∞	1	0	4	∞	∞	∞
5	8	11	13	4	0	6	∞	1
6	1	4	6	∞	6	0	5	∞
7	∞	1	∞	∞	∞	5	0	1
8	∞	∞	2	∞	1	∞	1	0

after k = 2:

	1	2	3	4	5	6	7	8
1	0	3	5	∞	8	1	4	∞
2	3	0	2	∞	11	4	1	∞
3	5	2	0	1	13	6	3	2
4	∞	∞	1	0	4	∞	∞	∞
5	8	11	13	4	0	6	12	1
6	1	4	6	∞	6	0	5	∞
7	4	1	3	∞	12	5	0	1
8	∞	∞	2	∞	1	∞	1	0

after k = 3:

	1	2	3	4	5	6	7	8
1	0	3	5	6	8	1	4	7
2	3	0	2	3	11	4	1	4
3	5	2	0	1	13	6	3	2
4	6	3	1	0	4	7	4	3
5	8	11	13	4	0	6	12	1
6	1	4	6	7	6	0	5	8
7	4	1	3	4	12	5	0	1
8	7	4	2	3	1	8	1	0

after k = 4:

	1	2	3	4	5	6	7	8
1	0	3	5	6	8	1	4	7
2	3	0	2	3	7	4	1	4
3	5	2	0	1	5	6	3	2
4	6	3	1	0	4	7	4	3
5	8	7	5	4	0	6	8	1
6	1	4	6	7	6	0	5	8
7	4	1	3	4	8	5	0	1
8	7	4	2	3	1	8	1	0

after k = 5:

	1	2	3	4	5	6	7	8
1	0	3	5	6	8	1	4	7
2	3	0	2	3	7	4	1	4
3	5	2	0	1	5	6	3	2
4	6	3	1	0	4	7	4	3
5	8	7	5	4	0	6	8	1
6	1	4	6	7	6	0	5	7
7	4	1	3	4	8	5	0	1
8	7	4	2	3	1	7	1	0

after k = 6:

	1	2	3	4	5	6	7	8
1	0	3	5	6	7	1	4	7
2	3	0	2	3	7	4	1	4
3	5	2	0	1	5	6	3	2
4	6	3	1	0	4	7	4	3
5	7	7	5	4	0	6	8	1
6	1	4	6	7	6	0	5	7
7	4	1	3	4	8	5	0	1
8	7	4	2	3	1	7	1	0

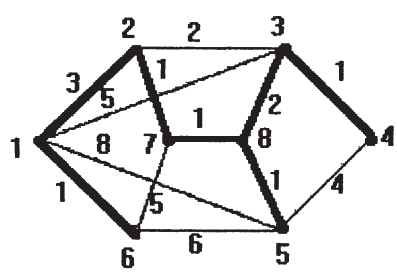
after k = 7:

	1	2	3	4	5	6	7	8
1	0	3	5	6	7	1	4	5
2	3	0	2	3	7	4	1	2
3	5	2	0	1	5	6	3	2
4	6	3	1	0	4	7	4	3
5	7	7	5	4	0	6	8	1
6	1	4	6	7	6	0	5	6
7	4	1	3	4	8	5	0	1
8	5	2	2	3	1	6	1	0

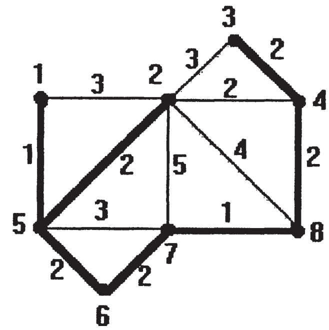
after k = 8:

	1	2	3	4	5	6	7	8
1	0	3	5	6	6	1	4	5
2	3	0	2	3	3	4	1	2
3	5	2	0	1	3	6	3	2
4	6	3	1	0	4	7	4	3
5	6	3	3	4	0	6	2	1
6	1	4	6	7	6	0	5	6
7	4	1	3	4	2	5	0	1
8	5	2	2	3	1	6	1	0

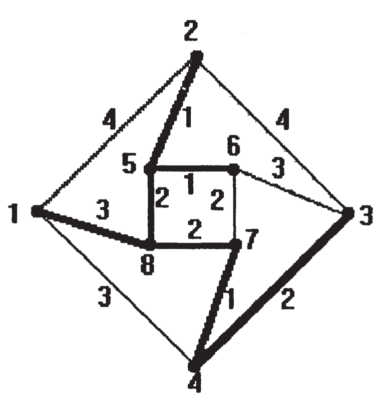
17. IN = {1,6,2,7,8,5,3,4}



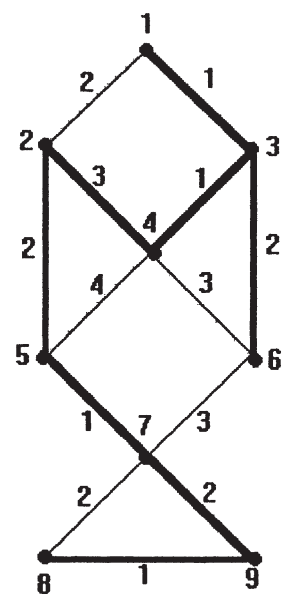
18. IN = {1,5,2,6,7,8,4,3}



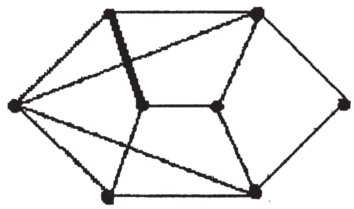
*19. IN = {1,8,5,6,2,7,4,3}



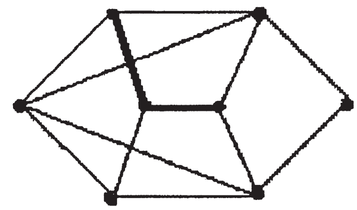
20. IN = {1,3,4,2,5,7,9,8,6}



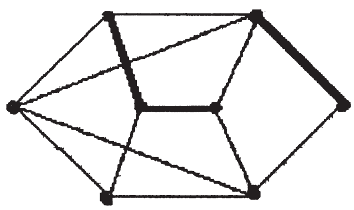
*21. Kruskal's algorithm develops a minimal spanning tree as follows.



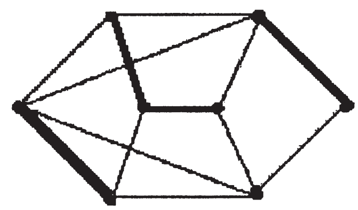
(1)



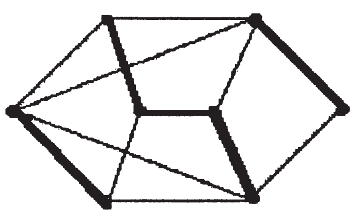
(2)



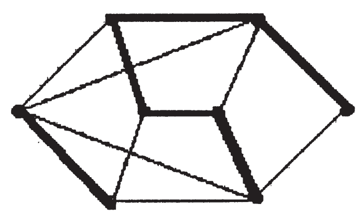
(3)



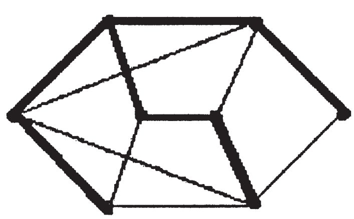
(4)



(5)



(6)



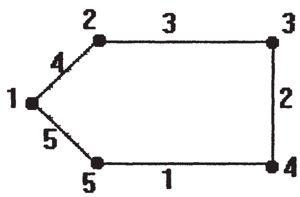
(7)

22. Result can agree with Exercise 16.

23. Result can agree with Exercise 17.

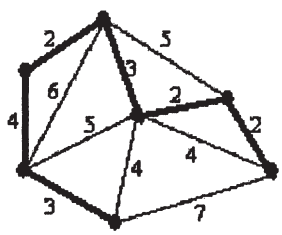
24. Result can agree with Exercise 18.

25.



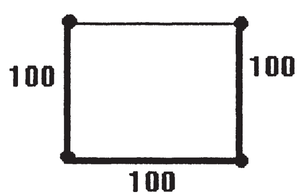
The shortest path from 1 to 5 is 1-5 with distance 5. If the algorithm added the node closest to IN at each step, it would choose the path 1-2-3-4-5 with distance 10.

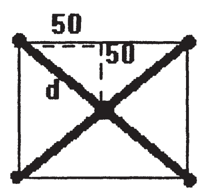
26. The solution is to find a minimal spanning tree for the graph, as shown here.



*27. a. weight = 300

b. $d = 50\sqrt{2} = 70.750 \quad 4d = 282.8$





28. Dijkstra's algorithm was $\Theta(n^2)$ in the worst case, which was when all nodes are brought into IN. This is the situation to find the distance from the start node to any other node. Repeating this process with all n nodes, in turn, as the start node would result in an algorithm of order $n\Theta(n^2) = \Theta(n^3)$. Algorithm AllPairsShortestPath is clearly $\Theta(n^3)$ because of the nested **for** loops. Therefore the algorithms are the same order of magnitude (although AllPairsShortestPath has the advantage of simplicity).

EXERCISES 6.4

*1. a b c e f d h g j i

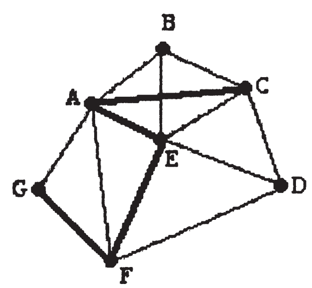
2. c a b e f d h g j i

3. d a b c e f h g j i

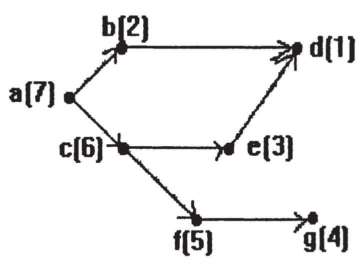
4. g b a c e f d h i j

*5. e b a c f d h g j i	6. h f c a b e g j i d
*7. a b c f j g d e h k i	8. e b a c f j g d h k i
*9 f c a b d e h k i g j	10. h e b a c f j g d i k
*11. a b c d e g f h j i	12. c a b e f d g h j i
13. d a f b c e h g i j	14. g b e h j a c f i d
15. e b c f g a d h j i	16. h f g i c d e b j a
*17. a b c d e f g h i j k	18. e b d h i a c k f g j
19. f c j a b g d e h i k	20. h e k b d i a c f g j
*21. a b c e g d f h	22. g d a b c e h f
23. f b	*24. a b c d e g h f
25. g d f h a e b c	26. f b

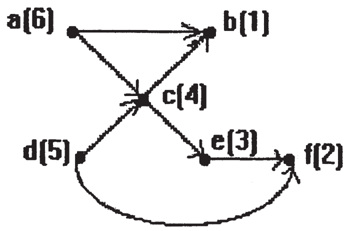
27. Begin a dfs at node C, using only the subgraph involving nodes C, A, E, F, and G. The arcs to previously unvisited nodes form the spanning tree.



*28. Begin a dfs at node a: a c f g e b d



29. Begin a dfs at node d, and then at node a: a d c e f b



30. Do a breadth-first search starting from the root and visiting adjacent nodes left to right.