Exercises 22.1-1
Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees?

Exercises 22.1-2
Given an adjacency-list representation for a complete binary tree on 7 vertices. Give an equivalent adjacency-matrix representation. Assume that vertices are numbered from 1 to 7 as in a binary heap.

Exercises 22.1-3
The transpose of a directed graph \( G = (V, E) \) is the graph \( G^T = (V, E^T) \), where \( E^T = \{(v, u) \in V \times V : (u, v) \in E\} \). Thus, \( G^T \) is \( G \) with all its edges reversed. Describe efficient algorithms for computing \( G^T \) from \( G \), for both the adjacency-list and adjacency-matrix representations of \( G \). Analyze the running times of your algorithms.

Exercises 22.1-4
Given an adjacency-list representation of a multigraph \( G = (V, E) \), describe an \( O(V + E) \)-time algorithm to compute the adjacency-list representation of the "equivalent" undirected graph \( G' = (V, E') \), where \( E' \) consists of the edges in \( E \) with all multiple edges between two vertices replaced by a single edge and with all self-loops removed.

Exercises 22.1-5
The square of a directed graph \( G = (V, E) \) is the graph \( G^2 = (V, E^2) \) such that \( (u, w) \in E^2 \) if and only if for some \( v \in V \), both \( (u, v) \in E \) and \( (v, w) \in E \). That is, \( G^2 \) contains an edge between \( u \) and \( w \) whenever \( G \) contains a path with exactly two edges between \( u \) and \( w \). Describe efficient algorithms for computing \( G^2 \) from \( G \) for both the adjacency-list and adjacency-matrix representations of \( G \). Analyze the running times of your algorithms.

Exercises 22.1-6
When an adjacency-matrix representation is used, most graph algorithms require time \( \Omega(V^2) \), but there are some exceptions. Show that determining whether a directed graph \( G \) contains a universal sink—a vertex with in-degree \( |V| - 1 \) and out-degree 0—can be determined in time \( O(V) \), given an adjacency matrix for \( G \).

Exercises 22.1-7
The incidence matrix of a directed graph \( G = (V, E) \) is a \( |V| \times |E| \) matrix \( B = (b_{ij}) \) such that
\[
b_{ij} = \begin{cases} 
-1 & \text{if edge } j \text{ leaves vertex } i , \\
1 & \text{if edge } j \text{ enters vertex } i , \\
0 & \text{otherwise} .
\end{cases}
\]
Describe what the entries of the matrix product \( BB^T \) represent, where \( B^T \) is the transpose of \( B \).

Exercises 22.2-1
Show the \( d \) and \( \pi \) values that result from running breadth-first search on the directed graph of Figure 22.2(a), using vertex 3 as the source.

Exercises 22.2-2
Show the \( d \) and \( \pi \) values that result from running breadth-first search on the undirected graph of Figure 22.3, using vertex \( u \) as the source.

Exercises 22.2-3
What is the running time of BFS if its input graph is represented by an adjacency matrix and the algorithm is modified to handle this form of input?
Exercises 22.2-4
Argue that in a breadth-first search, the value $d[u]$ assigned to a vertex $u$ is independent of the order in which the vertices in each adjacency list are given. Using Figure 22.3 as an example, show that the breadth-first tree computed by BFS can depend on the ordering within adjacency lists.

Exercises 22.2-5
Give an example of a directed graph $G = (V, E)$, a source vertex $s \in V$, and a set of tree edges $E_π \subseteq E$ such that for each vertex $v \in V$, the unique path in the graph $(V, E_π)$ from $s$ to $v$ is a shortest path in $G$, yet the set of edges $E_π$ cannot be produced by running BFS on $G$, no matter how the vertices are ordered in each adjacency list.

Exercises 22.2-6
There are two types of professional wrestlers: "good guys" and "bad guys." Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have $n$ professional wrestlers and we have a list of $r$ pairs of wrestlers for which there are rivalries. Give an $O(n + r)$-time algorithm that determines whether it is possible to designate some of the wrestlers as good guys and the remainder as bad guys such that each rivalry is between a good guy and a bad guy. If is it possible to perform such a designation, your algorithm should produce it.

Exercises 22.2-7: *
The diameter of a tree $T = (V, E)$ is given by
$$\max_{u, v \in V} \delta(u, v),$$
that is, the diameter is the largest of all shortest-path distances in the tree. Give an efficient algorithm to compute the diameter of a tree, and analyze the running time of your algorithm.

Exercises 22.2-8
Let $G = (V, E)$ be a connected, undirected graph. Give an $O(V + E)$-time algorithm to compute a path in $G$ that traverses each edge in $E$ exactly once in each direction. Describe how you can find your way out of a maze if you are given a large supply of pennies.

Exercises 22.3-1
Make a 3-by-3 chart with row and column labels WHITE, GRAY, and BLACK. In each cell $(i, j)$, indicate whether, at any point during a depth-first search of a directed graph, there can be an edge from a vertex of color $i$ to a vertex of color $j$. For each possible edge, indicate what edge types it can be. Make a second such chart for depth-first search of an undirected graph.

Exercises 22.3-2
Show how depth-first search works on the graph of Figure 22.6. Assume that the for loop of lines 5–7 of the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show the discovery and finishing times for each vertex, and show the classification of each edge.

Figure 22.6: A directed graph for use in Exercises 22.3-2 and 22.5-2.
Exercises 22.3-3
Show the parenthesis structure of the depth-first search shown in Figure 22.4.

Exercises 22.3-4
Show that edge \((u, v)\) is
1. a tree edge or forward edge if and only if \(d[u] < d[v] < f[v] < f[u]\),
2. a back edge if and only if \(d[v] < d[u] < f[u] < f[v]\), and
3. a cross edge if and only if \(d[v] < f[v] < d[u] < f[u]\).

Exercises 22.3-5
Show that in an undirected graph, classifying an edge \((u, v)\) as a tree edge or a back edge according to whether \((u, v)\) or \((v, u)\) is encountered first during the depth-first search is equivalent to classifying it according to the priority of types in the classification scheme.

Exercises 22.3-6
Rewrite the procedure DFS, using a stack to eliminate recursion.

Exercises 22.3-8
Give a counterexample to the conjecture that if there is a path from \(u\) to \(v\) in a directed graph \(G\), then any depth-first search must result in \(d[v] \leq f[u]\).

Exercises 22.3-10
Explain how a vertex \(u\) of a directed graph can end up in a depth-first tree containing only \(u\), even though \(u\) has both incoming and outgoing edges in \(G\).

Exercises 22.3-11
Show that a depth-first search of an undirected graph \(G\) can be used to identify the connected components of \(G\), and that the depth-first forest contains as many trees as \(G\) has connected components. More precisely, show how to modify depth-first search so that each vertex \(v\) is assigned an integer label \(cc[v]\) between 1 and \(k\), where \(k\) is the number of connected components of \(G\), such that \(cc[u] = cc[v]\) if and only if \(u\) and \(v\) are in the same connected component.

Exercises 22.4-3
Give an algorithm that determines whether or not a given undirected graph \(G = (V, E)\) contains a cycle. Your algorithm should run in \(O(V)\) time, independent of \(|E|\).

Exercises 22.4-5
Another way to perform topological sorting on a directed acyclic graph \(G = (V, E)\) is to repeatedly find a vertex of in-degree 0, output it, and remove it and all of its outgoing edges from the graph. Explain how to implement this idea so that it runs in time \(O(V + E)\). What happens to this algorithm if \(G\) has cycles?