22.4-2

Give a linear-time algorithm that takes as input a directed acyclic graph $G = (V, E)$ and two vertices s and t, and returns the number of paths from s to t in G. For example, in the directed acyclic graph of Figure 22.8, there are exactly four paths from vertex p to vertex v: pov, $poryv$, $posryv$, and $psryv$. (Your algorithm only needs to count the paths, not list them.)

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 $|E|$.

22.4-4

Prove or disprove: If a directed graph G contains cycles, then TOPOLOGICAL-SORT(G) produces a vertex ordering that minimizes the number of "bad" edges that are inconsistent with the ordering produced.

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?

22.5 Strongly connected components

We now consider a classic application of depth-first search: decomposing a directed graph into its strongly connected components. This section shows how to do this decomposition using two depth-first searches. Many algorithms that work with directed graphs begin with such a decomposition. After decomposition, the algorithm is run separately on each strongly connected component. The solutions are then combined according to the structure of connections between components.

Recall from Appendix B that a strongly connected component of a directed graph $G = (V, E)$ is a maximal set of vertices $C \subseteq V$ such that for every pair of vertices u and v in C, we have both $u \rightsquigarrow v$ and $v \rightsquigarrow u$; that is, vertices u and v are reachable from each other. Figure 22.9 shows an example.

Our algorithm for finding strongly connected components of a graph $G = (V, E)$ uses the transpose of G, which is defined in Exercise 22.1-3 to be the graph $G^T = (V, E^T)$, where $E^T = \{(u, v) : (v, u) \in E\}$. That is, E^T consists of the edges of G with their directions reversed. Given an adjacency-list representation of G, the time to create G^T is $O(V + E)$. It is interesting to observe that G and G^T have

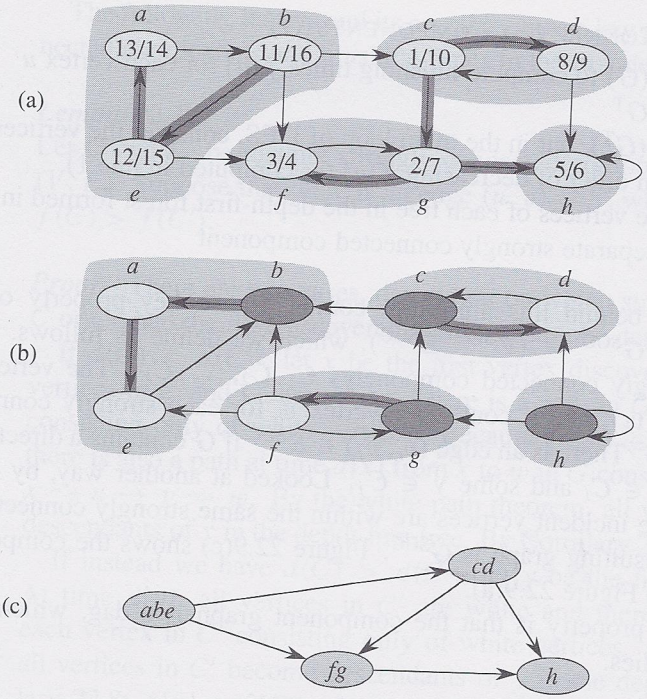

Figure 22.9 **(a)** A directed graph G. The strongly connected components of G are shown as shaded regions. Each vertex is labeled with its discovery and finishing times. Tree edges are shaded. **(b)** The graph G^T, the transpose of G. The depth-first forest computed in line 3 of STRONGLY-CONNECTED-COMPONENTS is shown, with tree edges shaded. Each strongly connected component corresponds to one depth-first tree. Vertices b, c, g, and h, which are heavily shaded, are the roots of the depth-first trees produced by the depth-first search of G^T. **(c)** The acyclic component graph G^{SCC} obtained by contracting all edges within each strongly connected component of G so that only a single vertex remains in each component.

exactly the same strongly connected components: u and v are reachable from each other in G if and only if they are reachable from each other in G^T. Figure 22.9(b) shows the transpose of the graph in Figure 22.9(a), with the strongly connected components shaded.

The following linear-time (i.e., $\Theta(V+E)$-time) algorithm computes the strongly connected components of a directed graph $G = (V, E)$ using two depth-first searches, one on G and one on G^T.

STRONGLY-CONNECTED-COMPONENTS(G)

1 call DFS(G) to compute finishing times $f[u]$ for each vertex u
2 compute G^T
3 call DFS(G^T), but in the main loop of DFS, consider the vertices in order of decreasing $f[u]$ (as computed in line 1)
4 output the vertices of each tree in the depth-first forest formed in line 3 as a separate strongly connected component

The idea behind this algorithm comes from a key property of the ***component graph*** $G^{SCC} = (V^{SCC}, E^{SCC})$, which we define as follows. Suppose that G has strongly connected components $C_1, C_2, \ldots, C_k$. The vertex set V^{SCC} is $\{v_1, v_2, \ldots, v_k\}$, and it contains a vertex v_i for each strongly connected component C_i of G. There is an edge $(v_i, v_j) \in E^{SCC}$ if G contains a directed edge (x, y) for some $x \in C_i$ and some $y \in C_j$. Looked at another way, by contracting all edges whose incident vertices are within the same strongly connected component of G, the resulting graph is G^{SCC}. Figure 22.9(c) shows the component graph of the graph in Figure 22.9(a).

The key property is that the component graph is a dag, which the following lemma implies.

Lemma 22.13

Let C and C' be distinct strongly connected components in directed graph $G = (V, E)$, let $u, v \in C$, let $u', v' \in C'$, and suppose that there is a path $u \rightsquigarrow u'$ in G. Then there cannot also be a path $v' \rightsquigarrow v$ in G.

Proof If there is a path $v' \rightsquigarrow v$ in G, then there are paths $u \rightsquigarrow u' \rightsquigarrow v'$ and $v' \rightsquigarrow v \rightsquigarrow u$ in G. Thus, u and v' are reachable from each other, thereby contradicting the assumption that C and C' are distinct strongly connected components. ■

We shall see that by considering vertices in the second depth-first search in decreasing order of the finishing times that were computed in the first depth-first search, we are, in essence, visiting the vertices of the component graph (each of which corresponds to a strongly connected component of G) in topologically sorted order.

Because STRONGLY-CONNECTED-COMPONENTS performs two depth-first searches, there is the potential for ambiguity when we discuss $d[u]$ or $f[u]$. In this section, these values always refer to the discovery and finishing times as computed by the first call of DFS, in line 1.

We extend the notation for discovery and finishing times to sets of vertices. If $U \subseteq V$, then we define $d(U) = \min_{u \in U} \{d[u]\}$ and $f(U) = \max_{u \in U} \{f[u]\}$. That is, $d(U)$ and $f(U)$ are the earliest discovery time and latest finishing time, respectively, of any vertex in U.

The following lemma and its corollary give a key property relating strongly connected components and finishing times in the first depth-first search.

Lemma 22.14
Let C and C' be distinct strongly connected components in directed graph $G = (V, E)$. Suppose that there is an edge $(u, v) \in E$, where $u \in C$ and $v \in C'$. Then $f(C) > f(C')$.

Proof There are two cases, depending on which strongly connected component, C or C', had the first discovered vertex during the depth-first search.

If $d(C) < d(C')$, let x be the first vertex discovered in C. At time $d[x]$, all vertices in C and C' are white. There is a path in G from x to each vertex in C consisting only of white vertices. Because $(u, v) \in E$, for any vertex $w \in C'$, there is also a path at time $d[x]$ from x to w in G consisting only of white vertices: $x \rightsquigarrow u \rightarrow v \rightsquigarrow w$. By the white-path theorem, all vertices in C and C' become descendants of x in the depth-first tree. By Corollary 22.8, $f[x] = f(C) > f(C')$.

If instead we have $d(C) > d(C')$, let y be the first vertex discovered in C'. At time $d[y]$, all vertices in C' are white and there is a path in G from y to each vertex in C' consisting only of white vertices. By the white-path theorem, all vertices in C' become descendants of y in the depth-first tree, and by Corollary 22.8, $f[y] = f(C')$. At time $d[y]$, all vertices in C are white. Since there is an edge (u, v) from C to C', Lemma 22.13 implies that there cannot be a path from C' to C. Hence, no vertex in C is reachable from y. At time $f[y]$, therefore, all vertices in C are still white. Thus, for any vertex $w \in C$, we have $f[w] > f[y]$, which implies that $f(C) > f(C')$. ■

The following corollary tells us that each edge in G^T that goes between different strongly connected components goes from a component with an earlier finishing time (in the first depth-first search) to a component with a later finishing time.

Corollary 22.15
Let C and C' be distinct strongly connected components in directed graph $G = (V, E)$. Suppose that there is an edge $(u, v) \in E^T$, where $u \in C$ and $v \in C'$. Then $f(C) < f(C')$.

Proof Since $(u, v) \in E^T$, we have $(v, u) \in E$. Since the strongly connected components of G and G^T are the same, Lemma 22.14 implies that $f(C) < f(C')$. ■

Corollary 22.15 provides the key to understanding why the STRONGLY-CONNECTED-COMPONENTS procedure works. Let us examine what happens when we perform the second depth-first search, which is on G^T. We start with the strongly connected component C whose finishing time $f(C)$ is maximum. The

search starts from some vertex $x \in C$, and it visits all vertices in C. By Corollary 22.15, there are no edges in G^T from C to any other strongly connected component, and so the search from x will not visit vertices in any other component. Thus, the tree rooted at x contains exactly the vertices of C. Having completed visiting all vertices in C, the search in line 3 selects as a root a vertex from some other strongly connected component C' whose finishing time $f(C')$ is maximum over all components other than C. Again, the search will visit all vertices in C', but by Corollary 22.15, the only edges in G^T from C' to any other component must be to C, which we have already visited. In general, when the depth-first search of G^T in line 3 visits any strongly connected component, any edges out of that component must be to components that were already visited. Each depth-first tree, therefore, will be exactly one strongly connected component. The following theorem formalizes this argument.

Theorem 22.16

STRONGLY-CONNECTED-COMPONENTS(G) correctly computes the strongly connected components of a directed graph G.

Proof We argue by induction on the number of depth-first trees found in the depth-first search of G^T in line 3 that the vertices of each tree form a strongly connected component. The inductive hypothesis is that the first k trees produced in line 3 are strongly connected components. The basis for the induction, when $k = 0$, is trivial.

In the inductive step, we assume that each of the first k depth-first trees produced in line 3 is a strongly connected component, and we consider the $(k + 1)$st tree produced. Let the root of this tree be vertex u, and let u be in strongly connected component C. Because of how we choose roots in the depth-first search in line 3, $f[u] = f(C) > f(C')$ for any strongly connected component C' other than C that has yet to be visited. By the inductive hypothesis, at the time that the search visits u, all other vertices of C are white. By the white-path theorem, therefore, all other vertices of C are descendants of u in its depth-first tree. Moreover, by the inductive hypothesis and by Corollary 22.15, any edges in G^T that leave C must be to strongly connected components that have already been visited. Thus, no vertex in any strongly connected component other than C will be a descendant of u during the depth-first search of G^T. Thus, the vertices of the depth-first tree in G^T that is rooted at u form exactly one strongly connected component, which completes the inductive step and the proof. ■

Here is another way to look at how the second depth-first search operates. Consider the component graph $(G^T)^{SCC}$ of G^T. If we map each strongly connected component visited in the second depth-first search to a vertex of $(G^T)^{SCC}$, the vertices of $(G^T)^{SCC}$ are visited in the reverse of a topologically sorted order. If we re-

verse the edges of $(G^T)^{SCC}$, we get the graph $((G^T)^{SCC})^T$. Because $((G^T)^{SCC})^T = G^{SCC}$ (see Exercise 22.5-4), the second depth-first search visits the vertices of G^{SCC} in topologically sorted order.

Exercises

22.5-1

How can the number of strongly connected components of a graph change if a new edge is added?

22.5-2

Show how the procedure STRONGLY-CONNECTED-COMPONENTS works on the graph of Figure 22.6. Specifically, show the finishing times computed in line 1 and the forest produced in line 3. Assume that the loop of lines 5–7 of DFS considers vertices in alphabetical order and that the adjacency lists are in alphabetical order.

22.5-3

Professor Deaver claims that the algorithm for strongly connected components can be simplified by using the original (instead of the transpose) graph in the second depth-first search and scanning the vertices in order of *increasing* finishing times. Is the professor correct?

22.5-4

Prove that for any directed graph G, we have $((G^T)^{SCC})^T = G^{SCC}$. That is, the transpose of the component graph of G^T is the same as the component graph of G.

22.5-5

Give an $O(V + E)$-time algorithm to compute the component graph of a directed graph $G = (V, E)$. Make sure that there is at most one edge between two vertices in the component graph your algorithm produces.

22.5-6

Given a directed graph $G = (V, E)$, explain how to create another graph $G' = (V, E')$ such that (a) G' has the same strongly connected components as G, (b) G' has the same component graph as G, and (c) E' is as small as possible. Describe a fast algorithm to compute G'.

22.5-7

A directed graph $G = (V, E)$ is said to be ***semiconnected*** if, for all pairs of vertices $u, v \in V$, we have $u \rightsquigarrow v$ or $v \rightsquigarrow u$. Give an efficient algorithm to determine whether or not G is semiconnected. Prove that your algorithm is correct, and analyze its running time.