20.4 Topological sort

This section shows how to use depth-first search to perform a topological sort of a directed acyclic graph, or a "dag" as it is sometimes called. A ***topological sort*** of a dag $G = (V, E)$ is a linear ordering of all its vertices such that if G contains an edge (u, v), then u appears before v in the ordering. Topological sorting is defined only on directed graphs that are acyclic; no linear ordering is possible when a directed graph contains a cycle. Think of a topological sort of a graph as an ordering of its vertices along a horizontal line so that all directed edges go from left to right. Topological sorting is thus different from the usual kind of "sorting" studied in Part II.

Many applications use directed acyclic graphs to indicate precedences among events. Figure 20.7 gives an example that arises when Professor Bumstead gets dressed in the morning. The professor must don certain garments before others (e.g., socks before shoes). Other items may be put on in any order (e.g., socks and pants). A directed edge (u, v) in the dag of Figure 20.7(a) indicates that garment u must be donned before garment v. A topological sort of this dag therefore gives a possible order for getting dressed. Figure 20.7(b) shows the topologically sorted dag as an ordering of vertices along a horizontal line such that all directed edges go from left to right.

The procedure TOPOLOGICAL-SORT topologically sorts a dag. Figure 20.7(b) shows how the topologically sorted vertices appear in reverse order of their finish times.

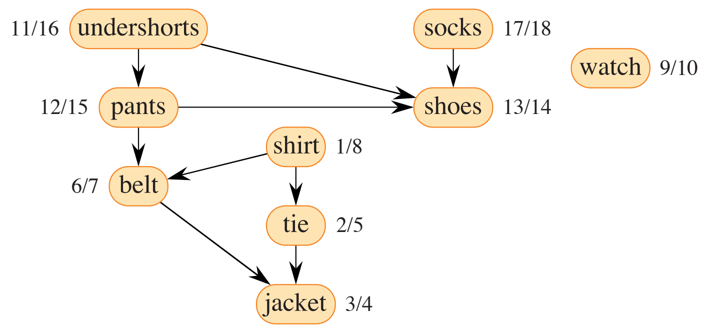
```
TOPOLOGICAL-SORT(G)
1  call DFS(G) to compute finish times v.f for each vertex v
2  as each vertex is finished, insert it onto the front of a linked list
3  return the linked list of vertices
```

The TOPOLOGICAL-SORT procedure runs in $\Theta(V + E)$ time, since depth-first search takes $\Theta(V + E)$ time and it takes $O(1)$ time to insert each of the $|V|$ vertices onto the front of the linked list.

To prove the correctness of this remarkably simple and efficient algorithm, we start with the following key lemma characterizing directed acyclic graphs.

Lemma 20.11
A directed graph G is acyclic if and only if a depth-first search of G yields no back edges.

(a)



(b)

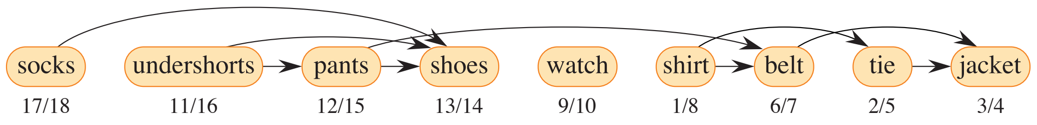


Figure 20.7 **(a)** Professor Bumstead topologically sorts his clothing when getting dressed. Each directed edge (u, v) means that garment u must be put on before garment v. The discovery and finish times from a depth-first search are shown next to each vertex. **(b)** The same graph shown topologically sorted, with its vertices arranged from left to right in order of decreasing finish time. All directed edges go from left to right.

Proof $\Rightarrow$: Suppose that a depth-first search produces a back edge (u, v). Then vertex v is an ancestor of vertex u in the depth-first forest. Thus, G contains a path from v to u, and the back edge (u, v) completes a cycle.

$\Leftarrow$: Suppose that G contains a cycle c. We show that a depth-first search of G yields a back edge. Let v be the first vertex to be discovered in c, and let (u, v) be the preceding edge in c. At time $v.d$, the vertices of c form a path of white vertices from v to u. By the white-path theorem, vertex u becomes a descendant of v in the depth-first forest. Therefore, (u, v) is a back edge. ■

Theorem 20.12
TOPOLOGICAL-SORT produces a topological sort of the directed acyclic graph provided as its input.

Proof Suppose that DFS is run on a given dag $G = (V, E)$ to determine finish times for its vertices. It suffices to show that for any pair of distinct vertices $u, v \in V$, if G contains an edge from u to v, then $v.f < u.f$. Consider any edge (u, v) explored by DFS(G). When this edge is explored, v cannot be gray, since then v would be an ancestor of u and (u, v) would be a back edge, contradicting Lemma 20.11. Therefore, v must be either white or black. If v is

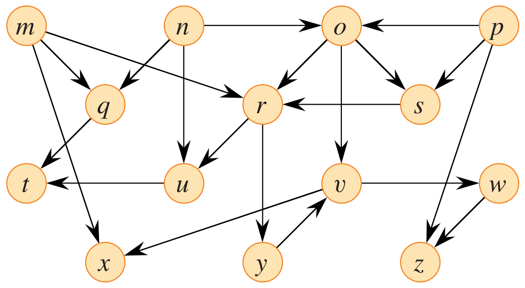


Figure 20.8 A dag for topological sorting.

white, it becomes a descendant of u, and so $v.f < u.f$. If v is black, it has already been finished, so that $v.f$ has already been set. Because the search is still exploring from u, it has yet to assign a timestamp to $u.f$, so that the timestamp eventually assigned to $u.f$ is greater than $v.f$. Thus, $v.f < u.f$ for any edge (u, v) in the dag, proving the theorem. ■

Exercises

20.4-1

Show the ordering of vertices produced by TOPOLOGICAL-SORT when it is run on the dag of Figure 20.8. 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.

20.4-2

Give a linear-time algorithm that, given a directed acyclic graph $G = (V, E)$ and two vertices $a, b \in V$, returns the number of simple paths from a to b in G. For example, the directed acyclic graph of Figure 20.8 contains exactly four simple paths from vertex p to vertex v: $\langle p, o, v \rangle$, $\langle p, o, r, y, v \rangle$, $\langle p, o, s, r, y, v \rangle$, and $\langle p, s, r, y, v \rangle$. Your algorithm needs only to count the simple paths, not list them.

20.4-3

Give an algorithm that determines whether an undirected graph $G = (V, E)$ contains a simple cycle. Your algorithm should run in $O(V)$ time, independent of $|E|$.

20.4-4

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