3 Topological sort (application of depth-first-search)

Consider a **directed acyclic graph**, which is a directed graph that has no cycles. If we have such a graph, it is possible to arrange the nodes in order, so that all of the edges point from left to right. This is called "topologically sorting" the graph.

Exercise: Why is it impossible to do this if the graph has a cycle?

As an example, directed acyclic graphs can be used to visualize dependencies between tasks. Maybe all your tasks involve putting on clothes, and there is an edge from your socks to your shoes because you have to put on your socks before putting on your shoes.

We can use a topological sort to figure out what order to do the tasks in. This guarantees that we don't try to put on shoes before socks.

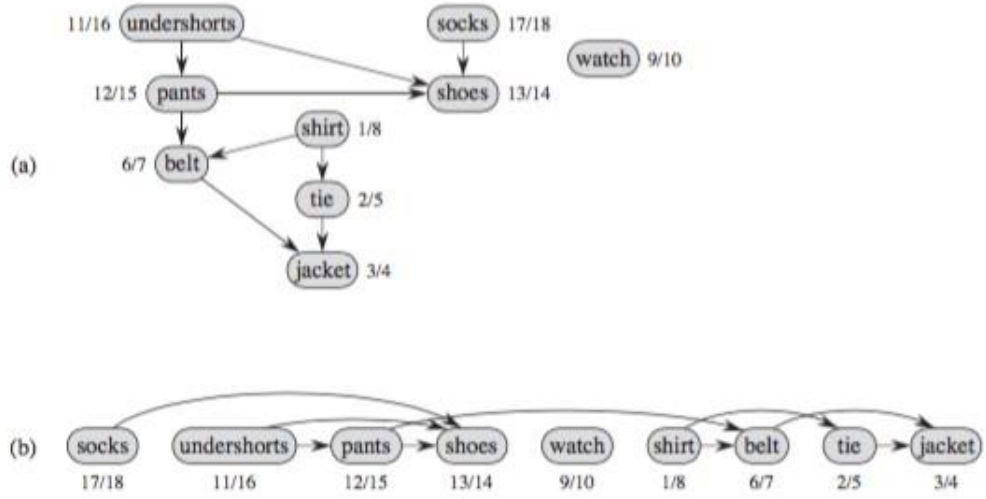

Figure 22.7 **(a)** Professor Bumstead topologically sorts his clothing when getting dressed. Each directed edge (u, v) means that garment u must be put on before garment v. The discovery and finishing times from a depth-first search are shown next to each vertex. **(b)** The same graph shown topologically sorted, with its vertices arranged from left to right in order of decreasing finishing time. All directed edges go from left to right.

Formally, a topological sort of a directed acyclic graph 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.

3.1 Algorithm

As it turns out, we can topologically sort a graph as follows:

TOPOLOGICAL-SORT(G)

1 call DFS(G) to compute finishing 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

This takes $\Theta(m + n)$ time, because depth-first-search takes $\Theta(m + n)$, and inserting into a linked list takes $O(1)$ time for each vertex.

3.2 Correctness

Suppose we run DFS on a graph. It suffices to show that for any pair of distinct vertices u,v, if the graph contains an edge from u to v, then $v.f < u.f$. This way, sorting the vertices in reverse order of finishing time will cause all of the edges to point in the correct direction.

There are three cases:

1. v is a descendant of u in the DFS tree. In this case, $v.f < u.f$ by the parenthesis property.
2. u is a descendant of v in the DFS tree. This is impossible because we assumed the graph had no cycles.
3. Neither vertex is a descendant of the other. In this case, the intervals $[u.d,u.f]$ and $[v.d,v.f]$ are disjoint, by the parenthesis property. However, we also know that $v.d < u.d$, because if u had been visited before v, then v would have been white when u was scanned, and by the white path theorem, it would become a descendant of u.
   Therefore, $v.f < u.f$.

AppliedRoots.com