# Topological Sort

Depth-first search (**DFS**) can be used to perform a topological sort of a directed acyclic graph, or a **DAG** as it is sometimes called. A **DAG** is a directed graph that cannot contain any cycle.

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. However, if the graph contains a cycle, then no linear ordering is possible. We can view 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" that you studied before.

Many applications use DAGs to indicate precedence among events. Consider the following problem-

**Example 1:**

*Absent-minded professor Bumstead has a problem when getting ready to go to work in the morning: he sometimes dresses out of order: he'll put his* ***shoes*** *on before putting the* ***socks*** *on, so he'll have to take the shoes off, put the socks on and then the shoes back on. There's also* ***shirt, tie, belt, shorts, pants, watch and jacket*** *that have to be put in a certain order. Can you help him?*

Notice that the professor must put on 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 1(a) indicates that garment $u$ must be donned before garment $v$. A topological sort of this DAG therefore gives an order for getting dressed. Figure 1(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 following simple algorithm topologically sorts a DAG:

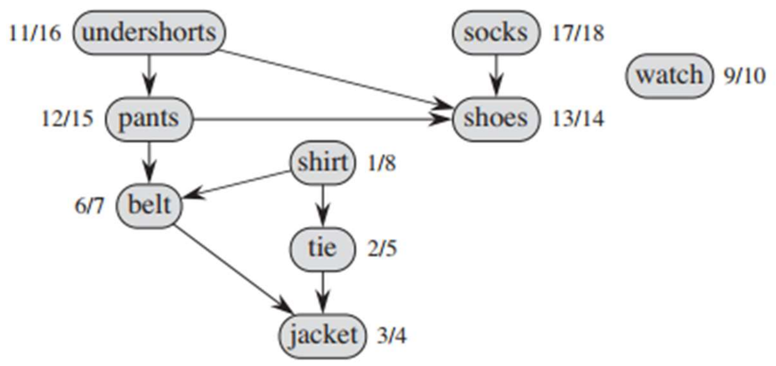
```
TOPOLOGICAL-SORT(G)

  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
```

*Figure 1(b)* shows how the topologically sorted vertices appear in reverse order of their finishing times.

**Time Complexity:** We can perform a topological sort in time $\theta(V + E)$, since DFS 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.

Observe that *a DAG can have multiple topologically sorted orders* depending on how the DFS is run.

(a) 

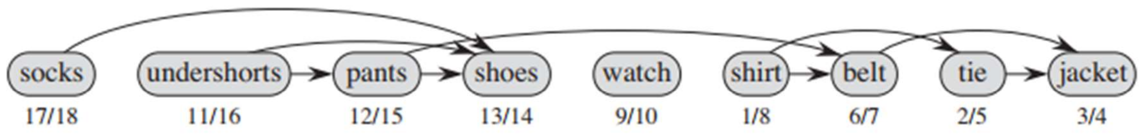
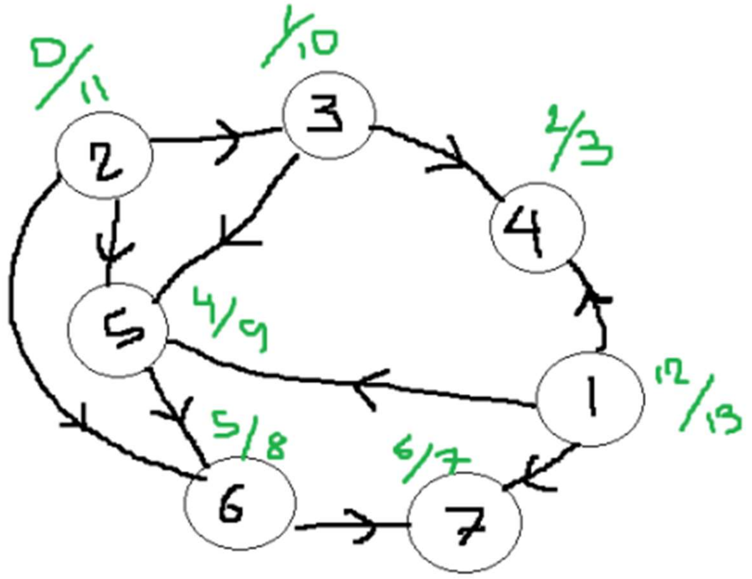
(b) 

Figure 1 **(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.

**Example 2:**

The graph below is a DAG with starting and ending times after we ran DFS on it.



If the ending times are arranged in a decreasing manner, we will get the ordering of vertices as shown below-

| **Ending times** | **13** | **11** | **10** | **9** | **8** | **7** | **3** |
|---|---|---|---|---|---|---|---|
| **Corresponding Vertices (Topologically sorted)** | **1** | **2** | **3** | **5** | **6** | **7** | **4** |