

## Application of DFS: Topological Sort

**Definition 1.** A *topological ordering* on the vertices is a total ordering assigning them numbers  $1, \dots, n$  such that only edges  $(i, j) \in E$  where  $i < j$  in the ordering.

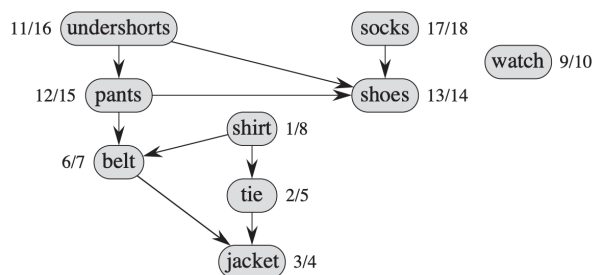


Figure 1: Top sort example graph from CLRS.

**Theorem 1.**  $G$  has a topological order  $\iff G$  is a DAG.

### Topological Sort Algorithm:

**Theorem 2.** If the tasks are scheduled by decreasing postorder number, then all precedence constraints are satisfied.

## Breadth-First Search

---

**Algorithm 1** BFS( $G, s$ )

---

**Input:** Graph  $G = (V, E)$  and starting vertex  $s$ .

initialize: (1) array  $dist$  of length  $n$ , (2) queue  $q$ , (3) linked list  $L$  of sets, (4) tree  $T = (\{s\}, \emptyset)$

$dist[s] = 0$

$L[0] = \{s\}$

enqueue  $s$  to  $q$

mark  $s$  as discovered and all other  $v$  as undiscovered

**while**  $size(q) > 0$  **do**

$v = dequeue(q)$

**for**  $(v, w) \in E$  **do**

**if**  $w$  is undiscovered **then**

            enqueue  $w$  in  $q$

            mark  $w$  as discovered

$dist(w) = dist(v) + 1$

            add  $w$  to  $L[dist(w)]$

            add  $(v, w)$  to  $T$

**end if**

**end for**

**end while**

**return**  $T, L$

---

What happens when we run BFS( $G, 1$ ) where  $G$  is the graph below?

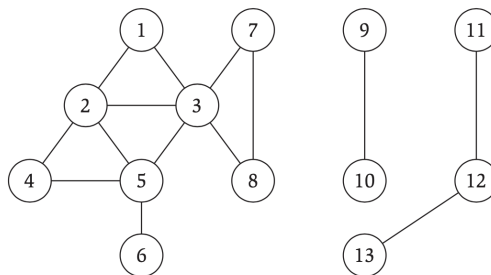


Figure 2: Example graph  $G$ . From Kleinberg Tardos.

What is BFS doing? BFS labels each vertex with the distance from  $s$ , or the number of edges in the shortest path from  $s$  to the vertex. (**Exercise:** Prove this!)

Runtime:

**Claim 1.** Let  $T$  be a breadth-first search tree, let  $x$  and  $y$  be nodes in  $T$  belonging to layers  $L_i$  and  $L_j$  respectively, and let  $(x, y)$  be an edge of  $G$ . Then  $i$  and  $j$  differ by at most 1.

*Proof.*