Book: Thomas H. Cormen
Section: 22.2, 22.3 and 22.4

- Breadth First Search
- Depth First Search
- Topological Sort

Graph

- $G = (V, E)$
  - V: Set of vertices
  - E: Set of edges
- Represented as
  - Adjacency List
  - Adjacency Matrix
- Weight, $w: E \rightarrow R$
- Edge Weight $w(u,v)$: Weight on edge u→v

Algorithm (Page - 595): BFS

```
BFS(G, s)
 1   for each vertex u ∈ G.V − {s}
 2       u.color = WHITE
 3       u.d = ∞
 4       u.π = NIL
 5   s.color = GRAY
 6   s.d = 0
 7   s.π = NIL
 8   Q = ∅
 9   ENQUEUE(Q, s)
10   while Q ≠ ∅
11       u = DEQUEUE(Q)
12       for each v ∈ G.Adj[u]
13           if v.color == WHITE
14               v.color = GRAY
15               v.d = u.d + 1
16               v.π = u
17               ENQUEUE(Q, v)
18       u.color = BLACK
```

Algorithm (Page - 604): DFS

```
DFS(G)
1  for each vertex u ∈ V[G]
2        do color[u] ← WHITE
3           π[u] ← NIL
4  time ← 0
5  for each vertex u ∈ V[G]
6        do if color[u] = WHITE
7              then DFS-VISIT(u)
```

```
DFS-VISIT(u)
1  color[u] ← GRAY          ▷ White vertex u has just been discovered.
2  time ← time + 1
3  d[u] ← time
4  for each v ∈ Adj[u]      ▷ Explore edge (u, v).
5       do if color[v] = WHITE
6             then π[v] ← u
7                  DFS-VISIT(v)
8  color[u] ← BLACK         ▷ Blacken u; it is finished.
9  f[u] ← time ← time + 1
```

Example (Page-613): Topological Sort

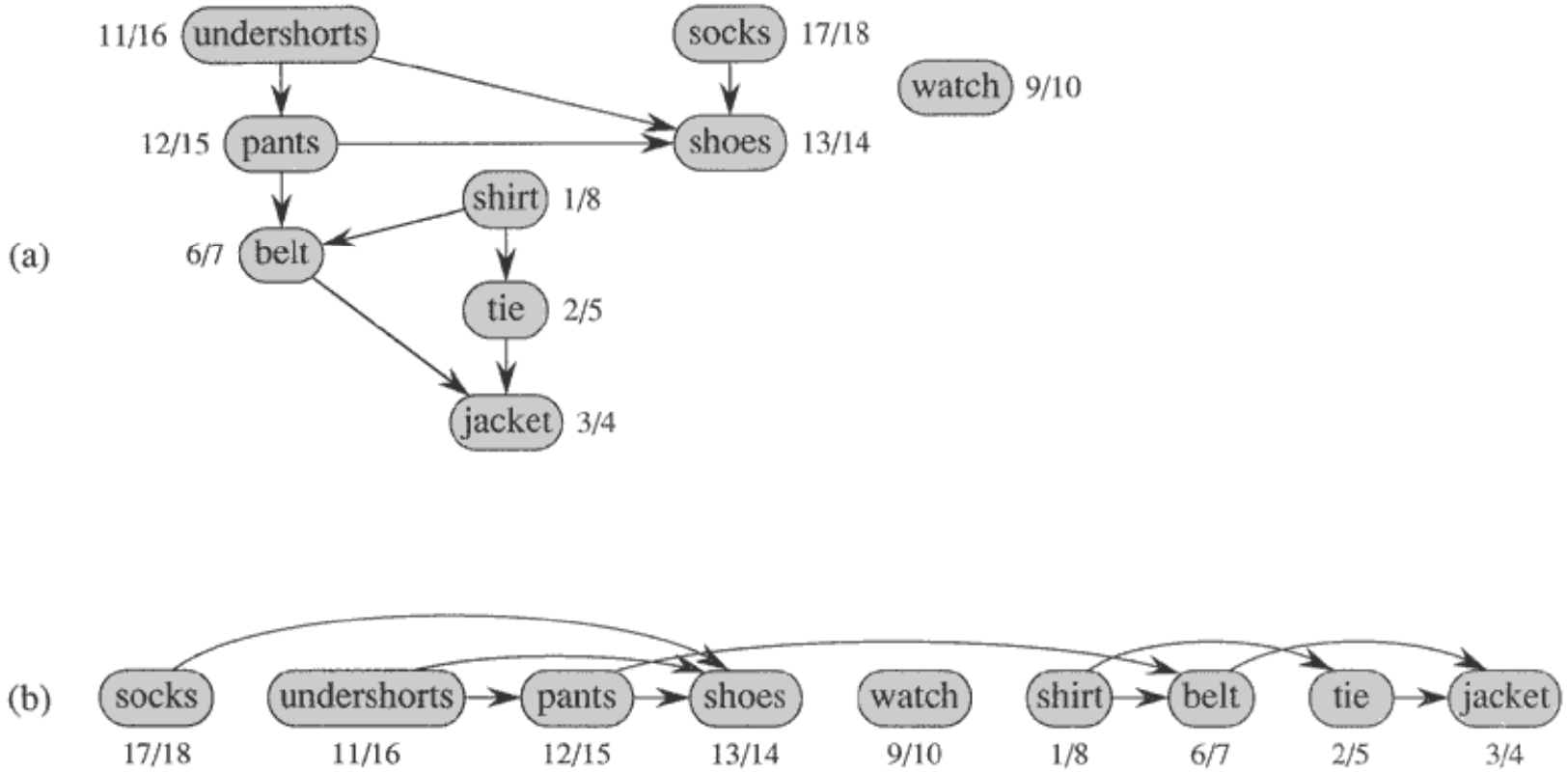

Figure 22.7 **(a)** Professor Bumstead topologically sorts his clothing when getting dressed. Each directed edge (u, v) means that garment u must be put on before garment v. The discovery and finishing times from a depth-first search are shown next to each vertex. **(b)** The same graph shown topologically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

Algorithm (Page - 613): Topological Sort

TOPOLOGICAL-SORT(G)

1 call DFS(G) to compute finishing times $f[v]$ 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

Thank You

Stay Safe