Topological Sort

CS-6th

Instructor: Dr. Ayesha Enayet

Topological Sort

- A topological sort of a dag (directed acyclic graph) 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.
- Only applicable on directed acyclic graphs.
- A directed graph G is acyclic if and only if a depth-first search of G yields no back edges.

Example

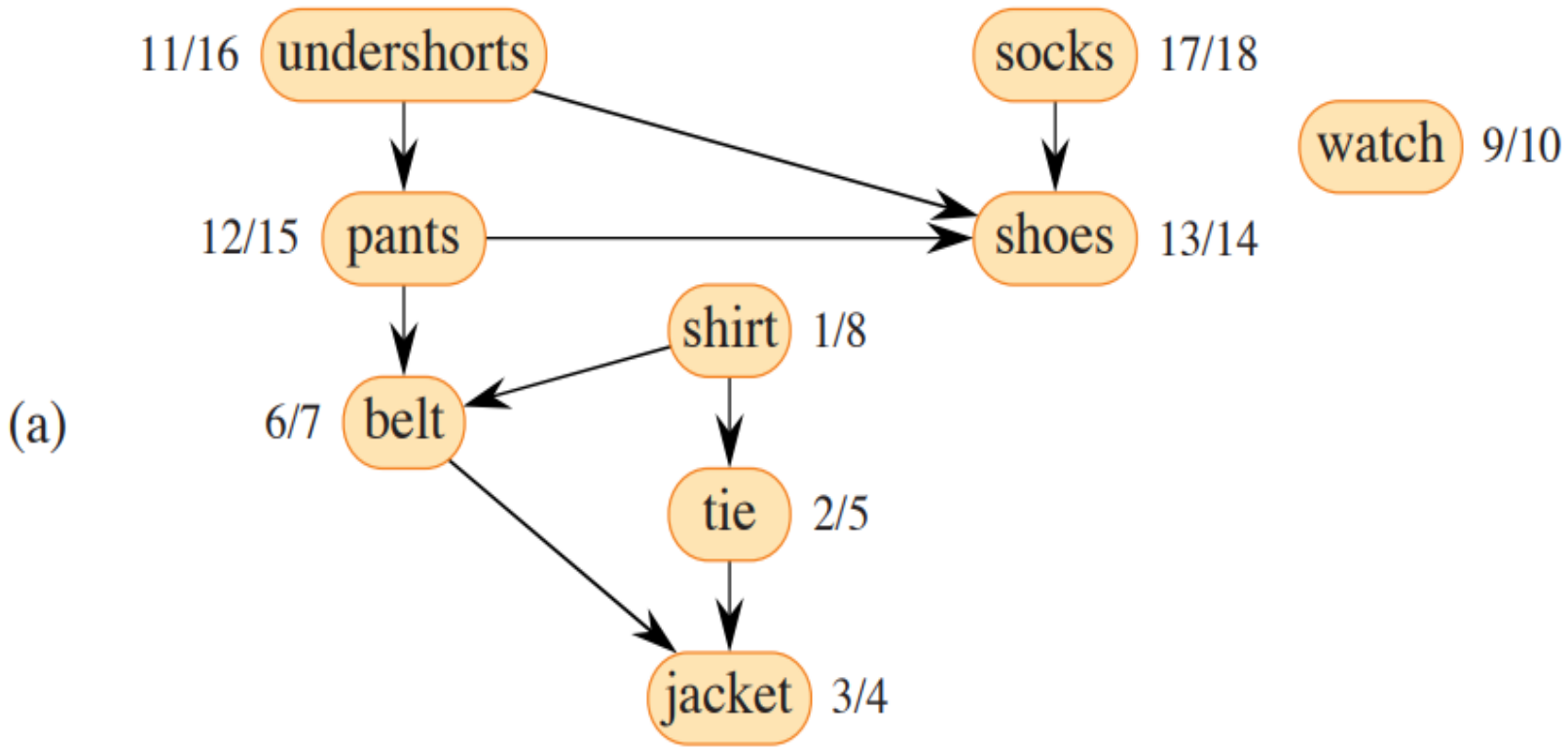
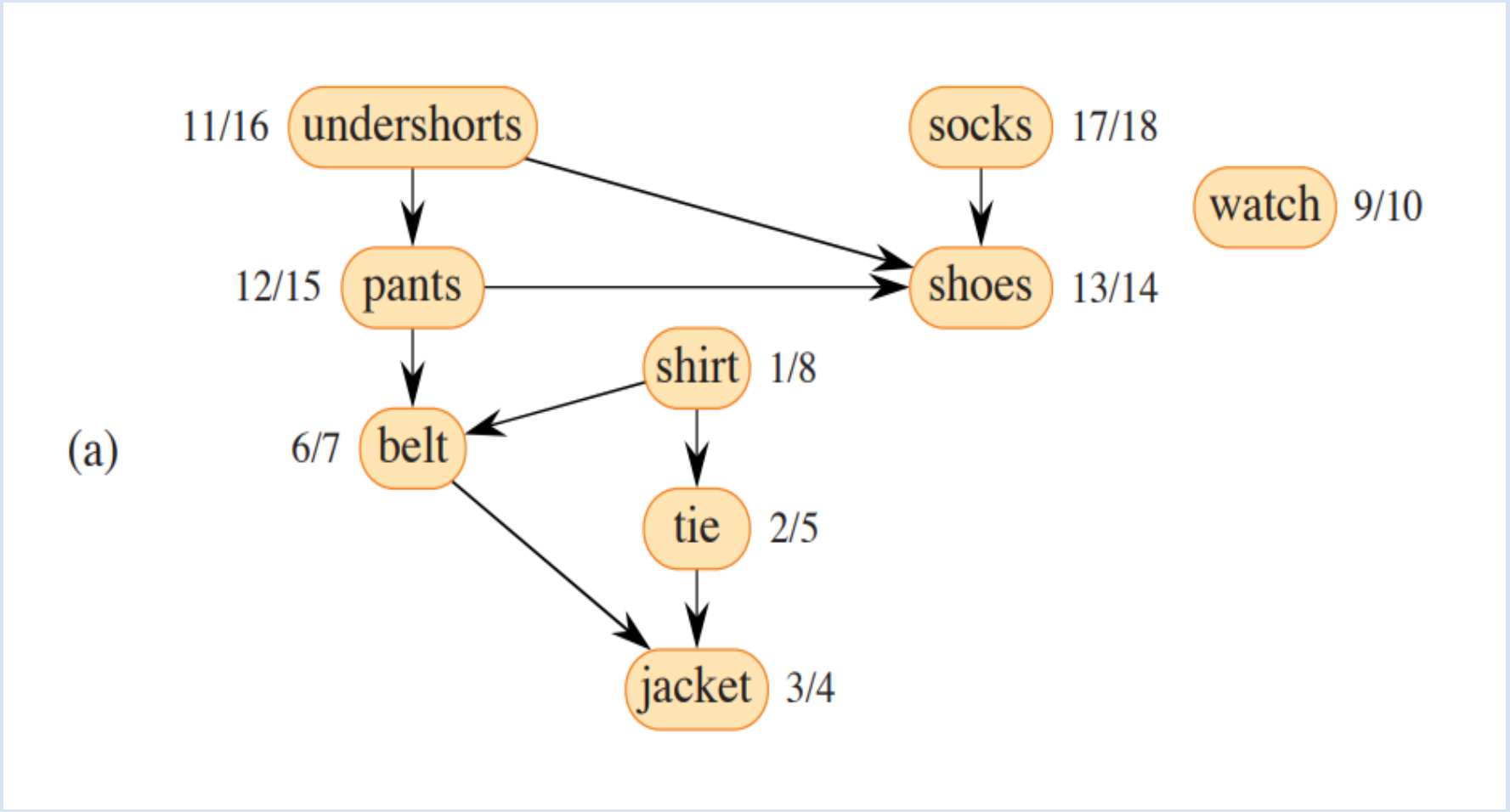
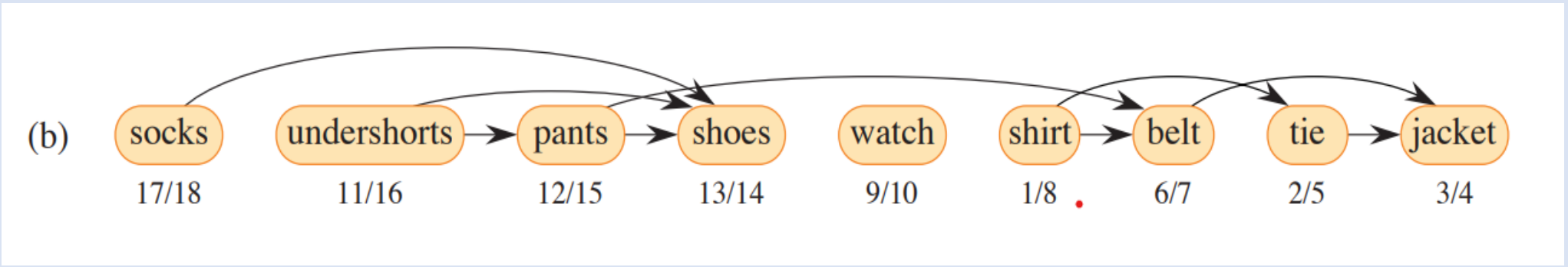


Figure 20.7 **(a)** Professor Bumstead topologically sorts his clothing when getting dressed. Each directed edge (u, v) means that garment u must be put on before garment v. The discovery and finish times from a depth-first search are shown next to each vertex. **(b)** The same graph shown topologically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

Input



Output



Topological Sort Algorithm

```
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
```

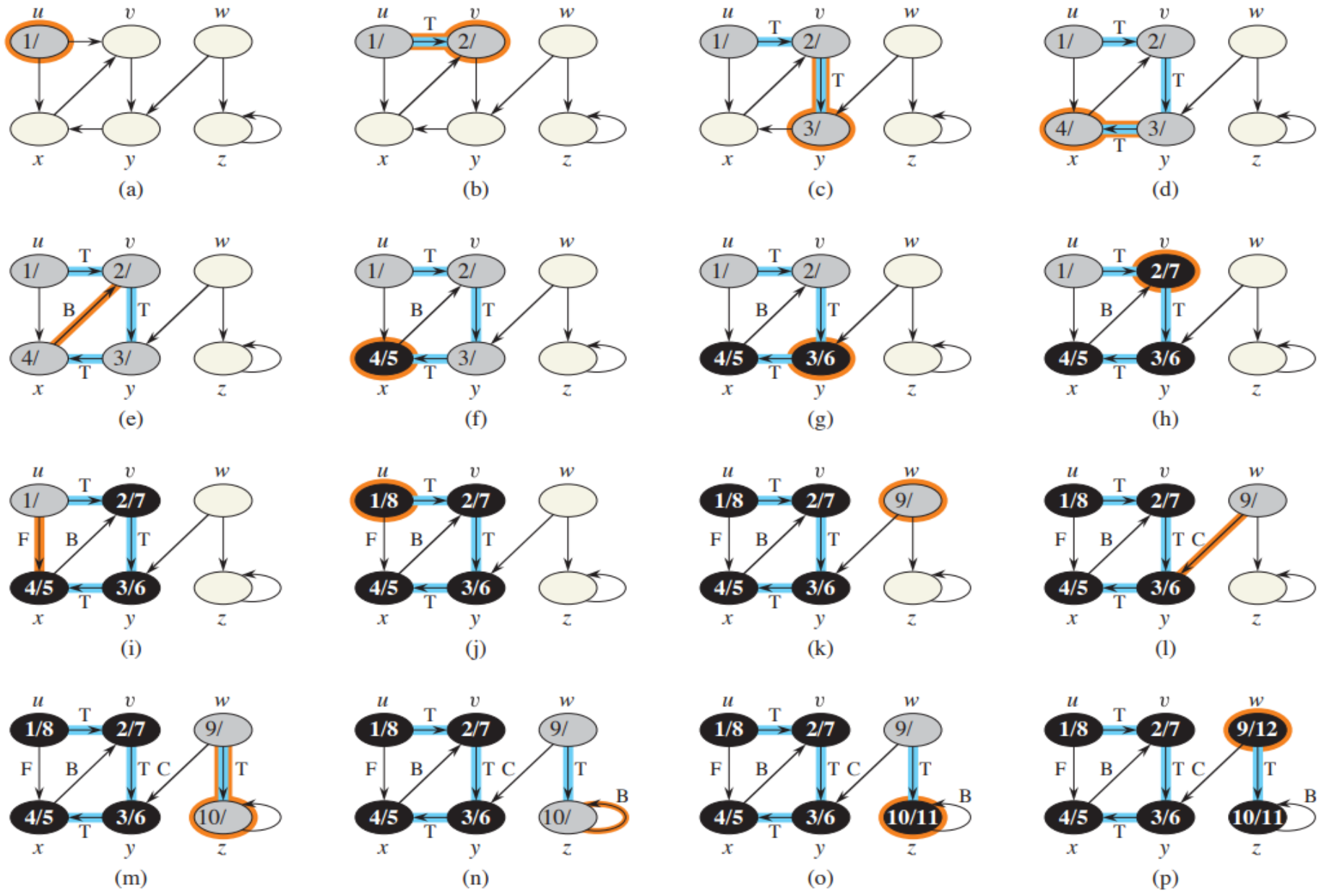
Time Complexity

- Θ (V+E)

Depth-First Search Algorithm

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

DFS-VISIT(G, u)
 1  time = time + 1                  // white vertex u has just been discovered
 2  u.d = time
 3  u.color = GRAY
 4  for each vertex v in G.Adj[u]    // explore each edge (u, v)
 5      if v.color == WHITE
 6          v.π = u
 7          DFS-VISIT(G, v)
 8  time = time + 1
 9  u.f = time
10  u.color = BLACK                  // blacken u; it is finished
```


u v w x y z
1/
2/
3/
4/
10/
9/12
4/5
3/6
2/7
1/8
10/11
T B F C
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
(k)
(l)
(m)
(n)
(o)
(p)