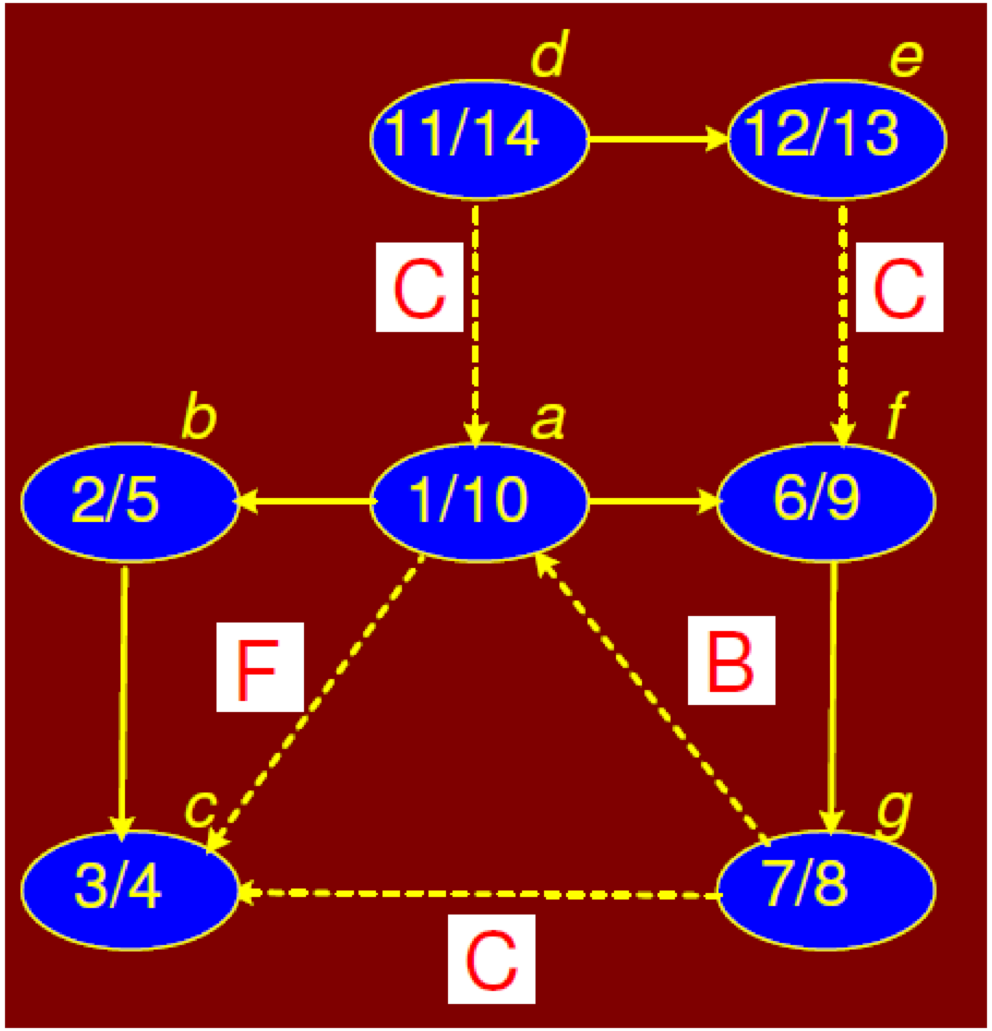
Graphs - Cycles, Topological Sorting

(Class 32)

From Book's Page Number 573 (Chapter 20)

DFS – Cycles



- The time stamps given by DFS allow us to determine a number of things about a graph or digraph.
- For example, we can determine whether the graph contains any *cycles*.
- We do this with the help of the following two theorems:

Theorem 1

- Given a digraph $G = (V, E)$, consider any DFS forest of G and consider any edge $(u, v) \in E$.
- If this edge is a tree, forward or cross edge, then $f[u] > f[v]$.
- If this edge is a back edge, then $f[u] \leq f[v]$.

Proof

- For the non-tree forward and back edges the proof follows directly from the parenthesis theorem.
- For example, for a forward edge (u, v), v is a descendent of u and so v's start-finish interval is contained within u's implying that v has an earlier finish time.
- For a cross edge (u, v) we know that these two time intervals are disjoint.
- When we were processing u, v was not white (otherwise (u, v) would be a tree edge), implying that v was started before u.
- Because the intervals are disjoint, v must have also finished before u.

Theorem 2

- Consider a digraph $G = (V, E)$ and any DFS forest for G.
- G has a cycle if and only if the DFS forest has a *back* edge.

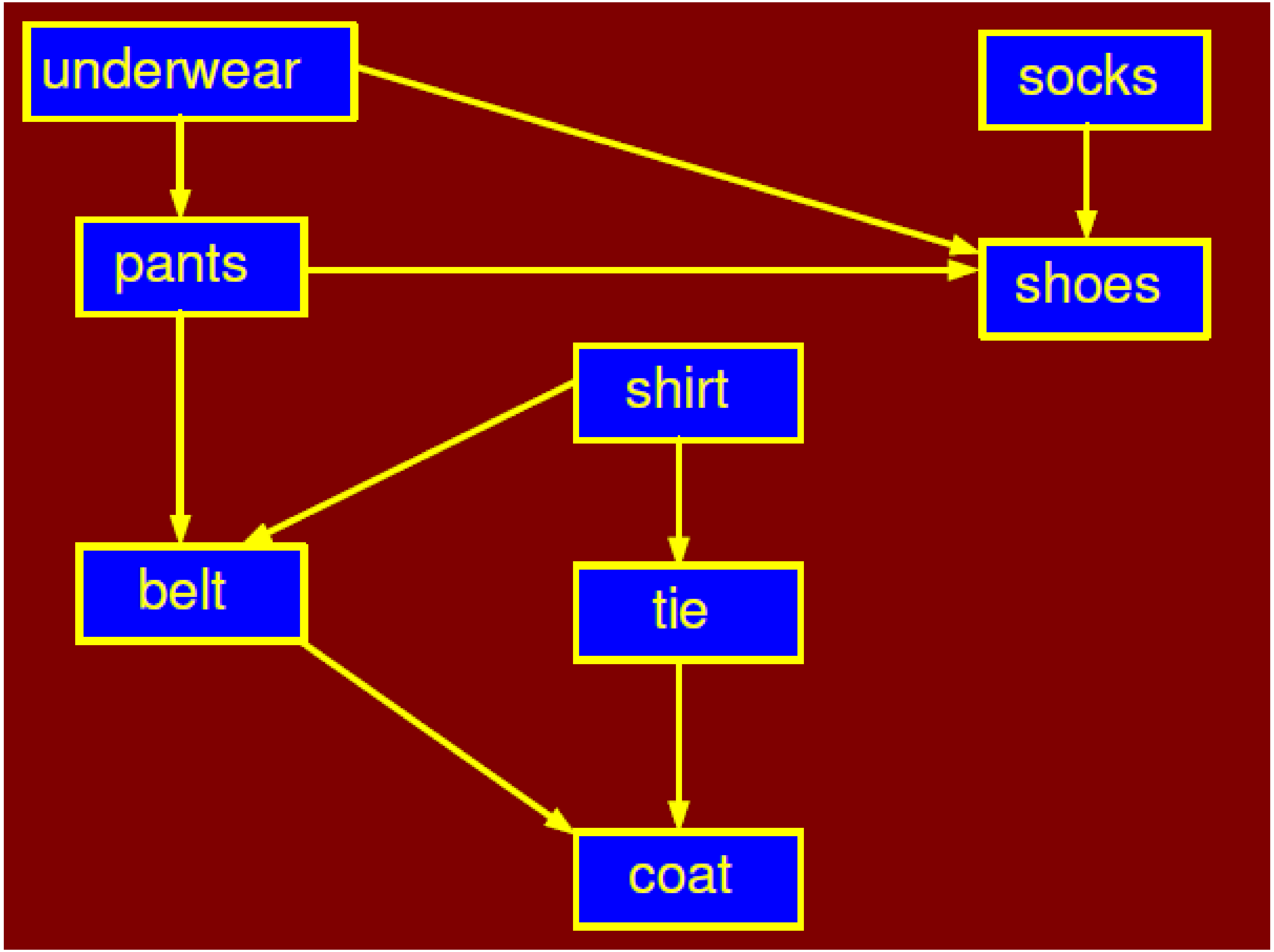
Proof

- If there is a back edge (u, v) then v is an ancestor of u and by following tree edge from v to u, we get a cycle.
- **Beware:**
  - No back edges means no cycles.
  - But you should not assume that there is some simple relationship between the number of back edges and the number of cycles.
  - For example, a DFS tree may only have a single back edge, and there may anywhere from one up to an exponential number of simple cycles in the graph.
  - A similar theorem applies to undirected graphs and is not hard to prove.

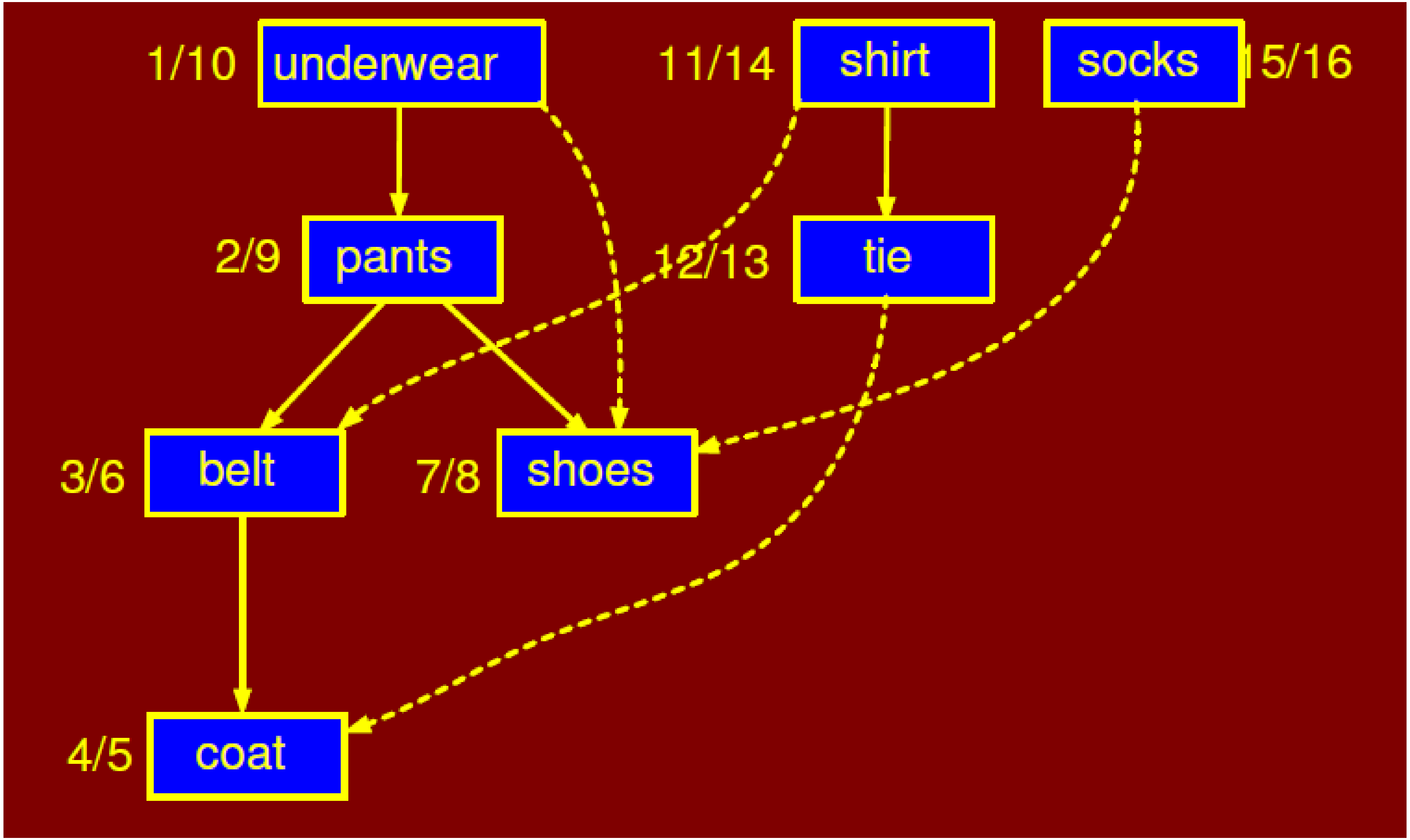
Precedence Constraint Graph

- A *directed acyclic graph* (DAG) arises in many applications where there is precedence or ordering constraints.
- There are a series of tasks to be performed and certain tasks must precede other tasks.
- For example, in construction, you have to build the first floor before the second floor, but you can do electrical work while doors and windows are being installed.

- In general, a *precedence constraint graph* is a DAG in which vertices are tasks and the edge (u, v) means that task u must be completed before task v begins.
- For example, consider the sequence followed when one wants to dress up in a suit.
- One possible order and its DAG are shown in the following figure.



Order of dressing up in a suit



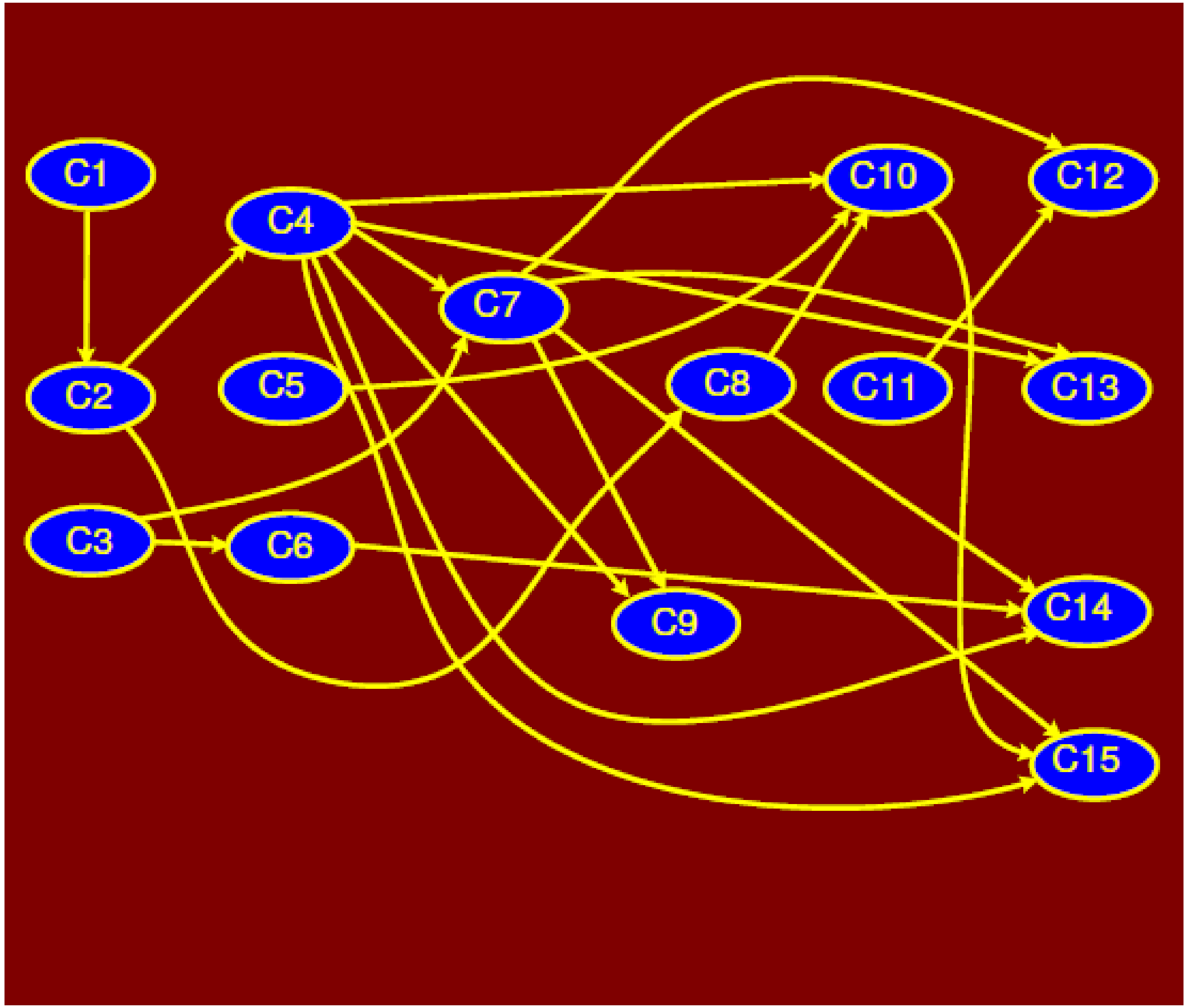
DFS of dressing up DAG with time stamps

- Another example of precedence constraint graph is the sets of prerequisites for CS courses in a typical undergraduate program.

C1	Introduction to Computers	
C2	Introduction to Computer Programming	
C3	Discrete Mathematics	
C4	Data Structures	C2
C5	Digital Logic Design	C2
C6	Automata Theory	C3
C7	Analysis of Algorithms	C3, C4
C8	Computer Organization and Assembly	C2
C9	Data Base Systems	C4, C7
C10	Computer Architecture	C4, C5,C8
C11	Computer Graphics	C4,C7
C12	Software Engineering	C7,C11
C13	Operating System	C4,C7,C11
C14	Compiler Construction	C4,C6,C8
C15	Computer Networks	C4,C7,C10

Prerequisites for CS courses

- The prerequisites can be represented with a precedence constraint graph:



Topological Sort

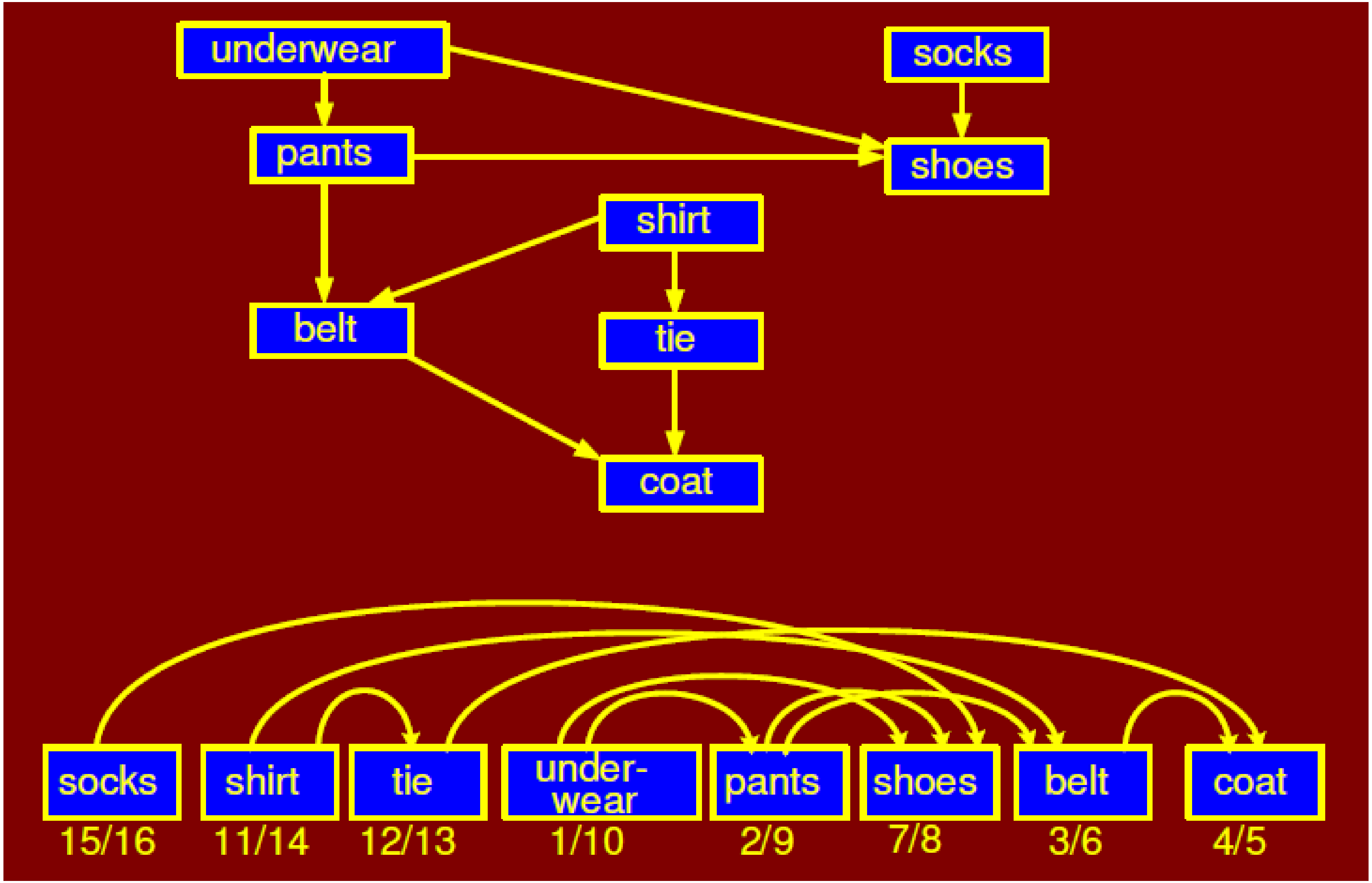
- A topological sort of a DAG is a linear ordering of the vertices of the DAG such that for each edge (u, v), u appears before v in the ordering.
- Computing a topological ordering is actually quite easy, given a DFS of the DAG.

- For every edge (u, v) in a DAG, the finish time of u is greater than the finish time of v (by the theorem).
- Thus, it would be sufficient to output the vertices in the reverse order of finish times.
- We run DFS on the DAG and when each vertex is finished, we add it to the front of a link-list.
- Note that in general, there may be many legal topological orders for a given DAG.

```
TOPOLOGICALSORT(G)
for (each u ∈ V)
    color[u] ← white
L ← new LinkedList()
for each u ∈ V
    if (color[u] = white)
        then TOPVISIT(u)
return L
```

```
TOPVISIT(u)
color[u] ← gray     // mark u visited
for (each v ∈ Adj[u])
    if (color[v] = white)
        then TOPVISIT(v)
Append u to the front of L
```

- The figure below shows the linear order obtained by the topological sort of the sequence of putting on a suit.
- The DAG is still the same; it is only that the order in which the vertices of the graph have been laid out is special.
- As a result, all directed edges go from left to right.



Topological sort of the dressing up sequence

Running Time Complexity

- This is a typical example of how DFS is used in applications.
- The running time is $O(V + E)$.