Graphs: Topological Sort

Bijendra Nath Jain

bnjain@iiitd.ac.in

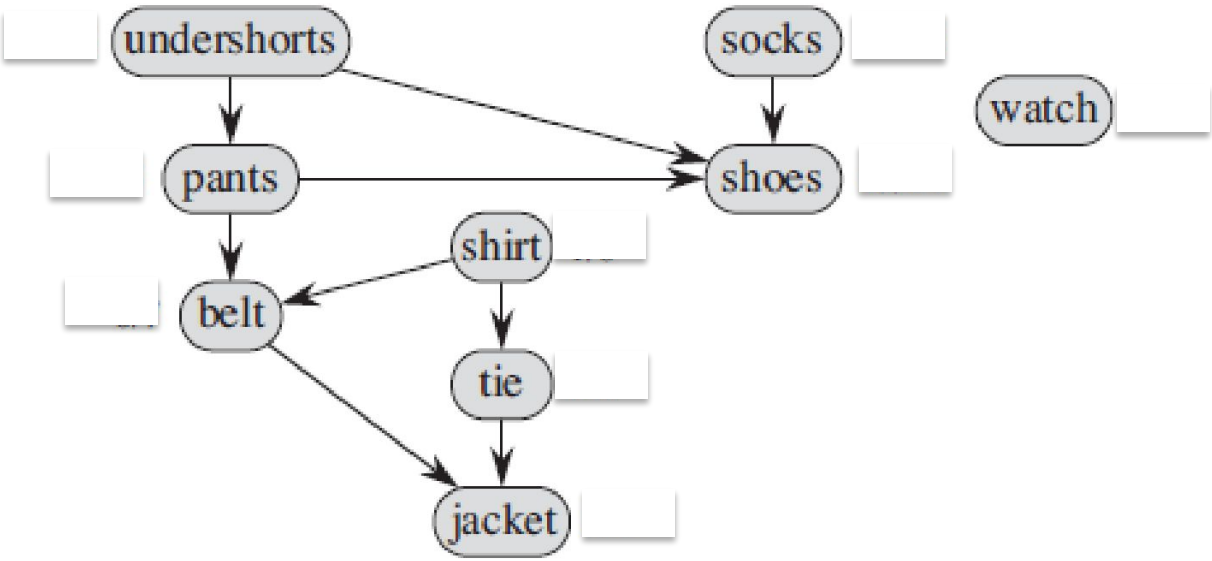
Some of the slides are from https://courses.cs.washington.edu/courses/cse373/22sp/

Outline

- Graphs:
  - Undirected graphs
  - Directed graphs
  - (Directed) acyclic graphs (or DAGs)
  - Sparse graphs
  - Weighted graphs
- Graph applications
- Representation of graphs:
  - Adjacency matrix
  - Linked lists
- Algorithms:
  - Traversal algorithms:
    - BFS
    - DFS
  - Topological sort
  - Minimum spanning trees
  - Dijkstra's Shortest path
    - One-to-one
    - One-to-many
    - Many-to-many

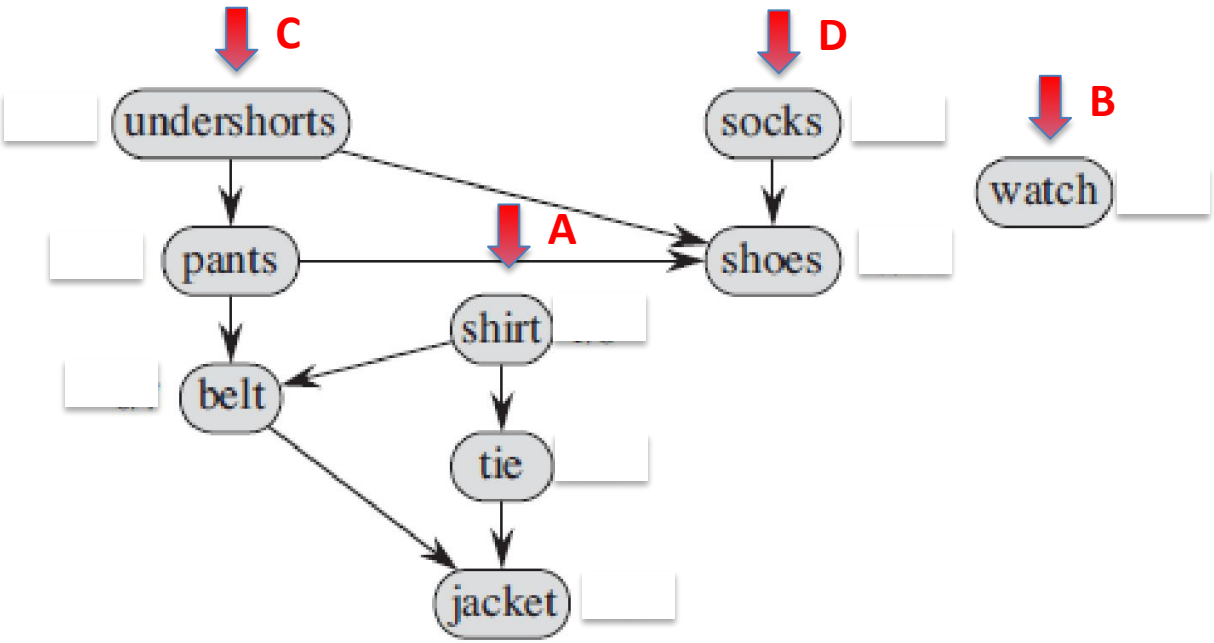
DFS, and topological-sort

- Consider running DFS on graph below



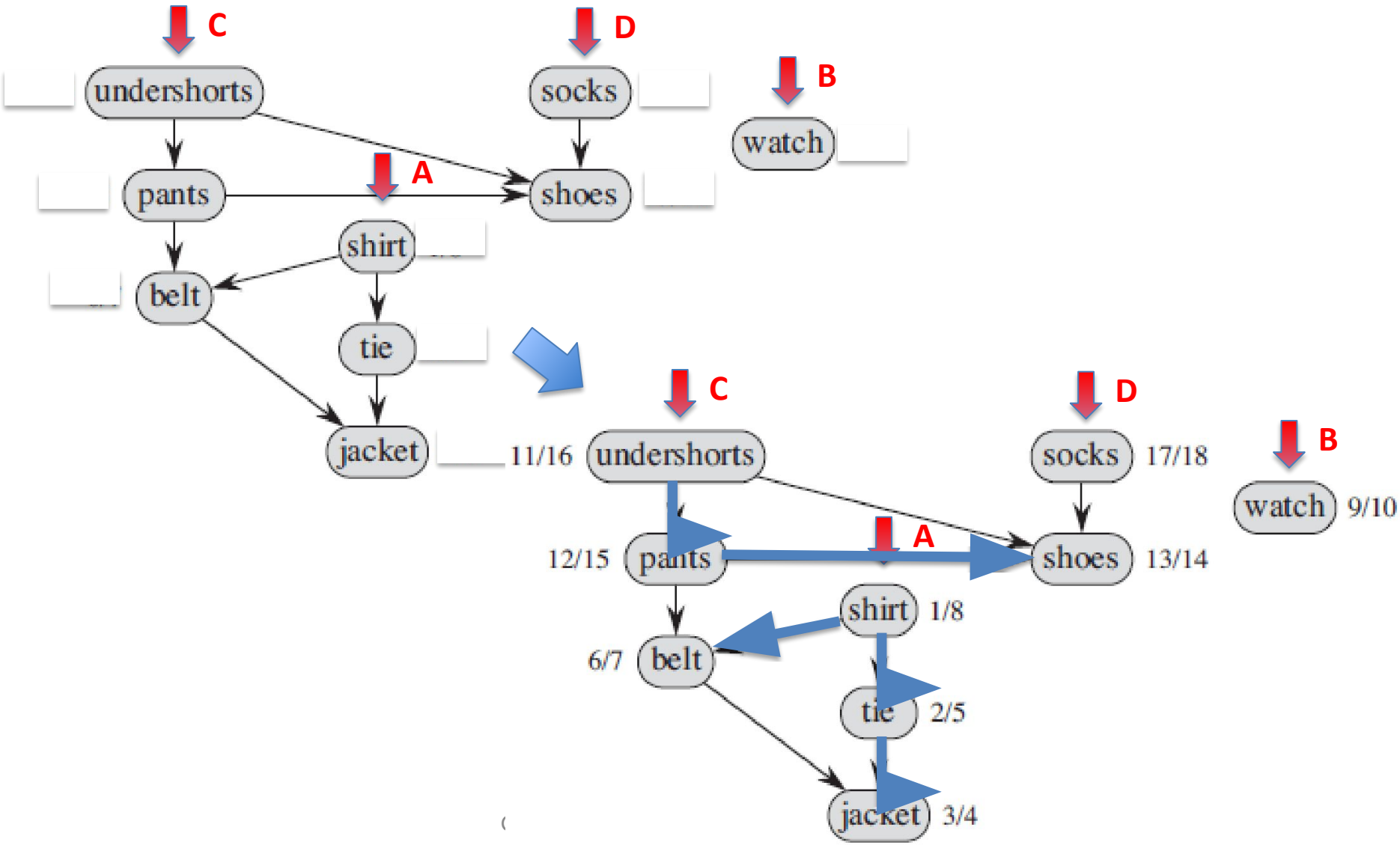
DFS, and topological-sort

- Consider running DFS on graph below
  - DFS-Visit is run from vertices A, B, C, D in that order



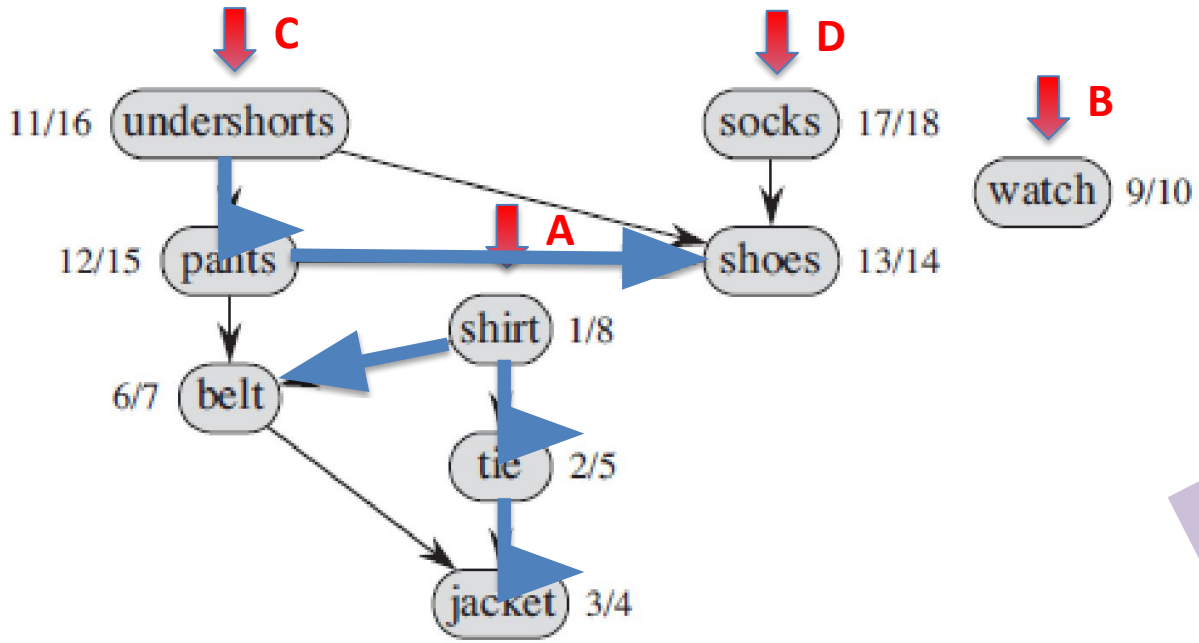
DFS, and topological-sort

- Consider running DFS on graph below
  - DFS-Visit is run from vertices A, B, C, D in that order



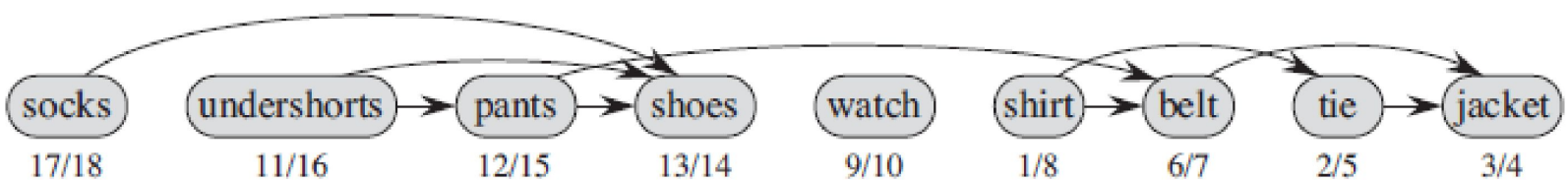
DFS, and topological-sort

- Consider running DFS on graph below
  - DFS-Visit is run from vertices A, B, C, D in that order



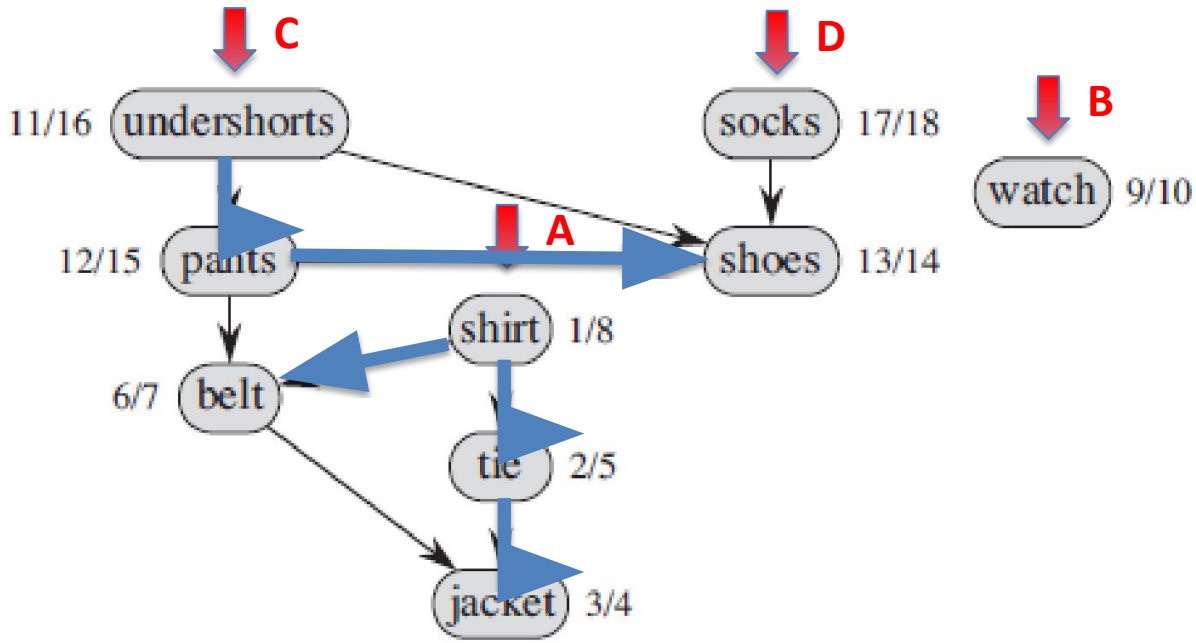
Any linear ordering of vertices in which all the arrows go to the right is a valid solution

  - One sequence in which garments may be worn without violating the "pre-requisite" requirements



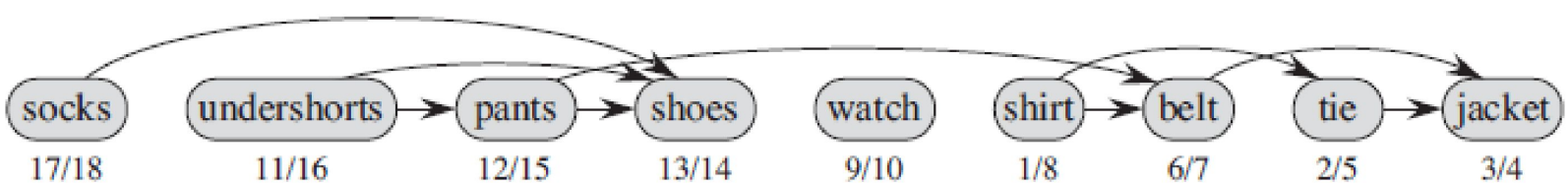
DFS, and topological-sort

- Consider running DFS on graph below
  - DFS-Visit is run from vertices A, B, C, D in that order



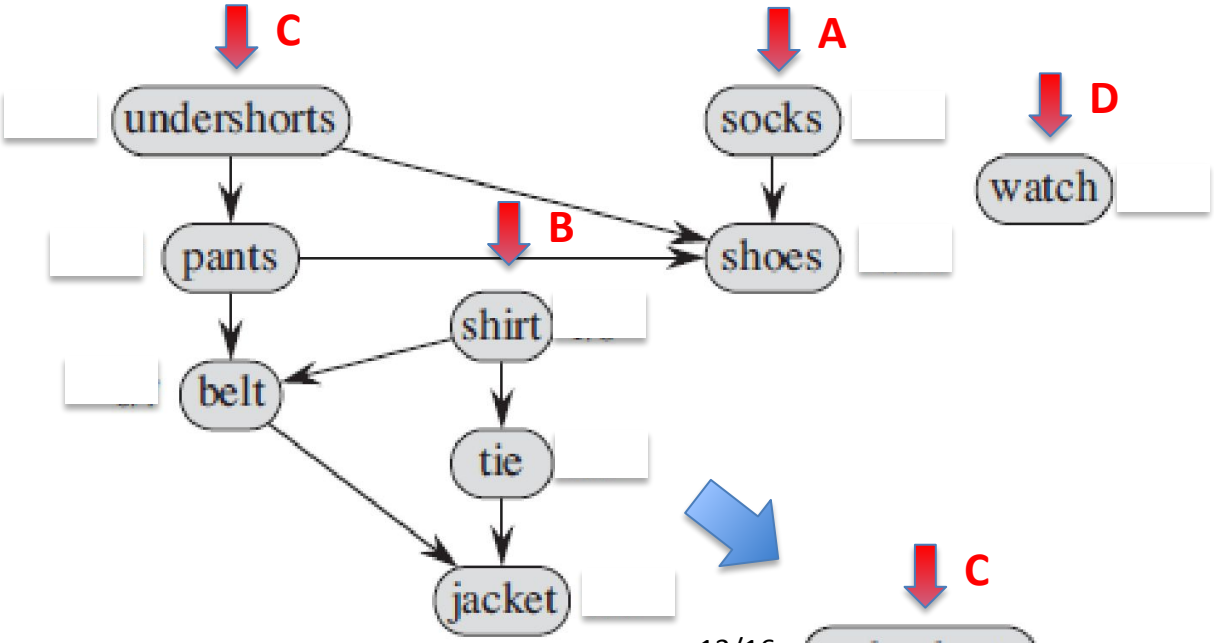
Choice of sequence of vertices from which DFS-Visit was run was arbitrary

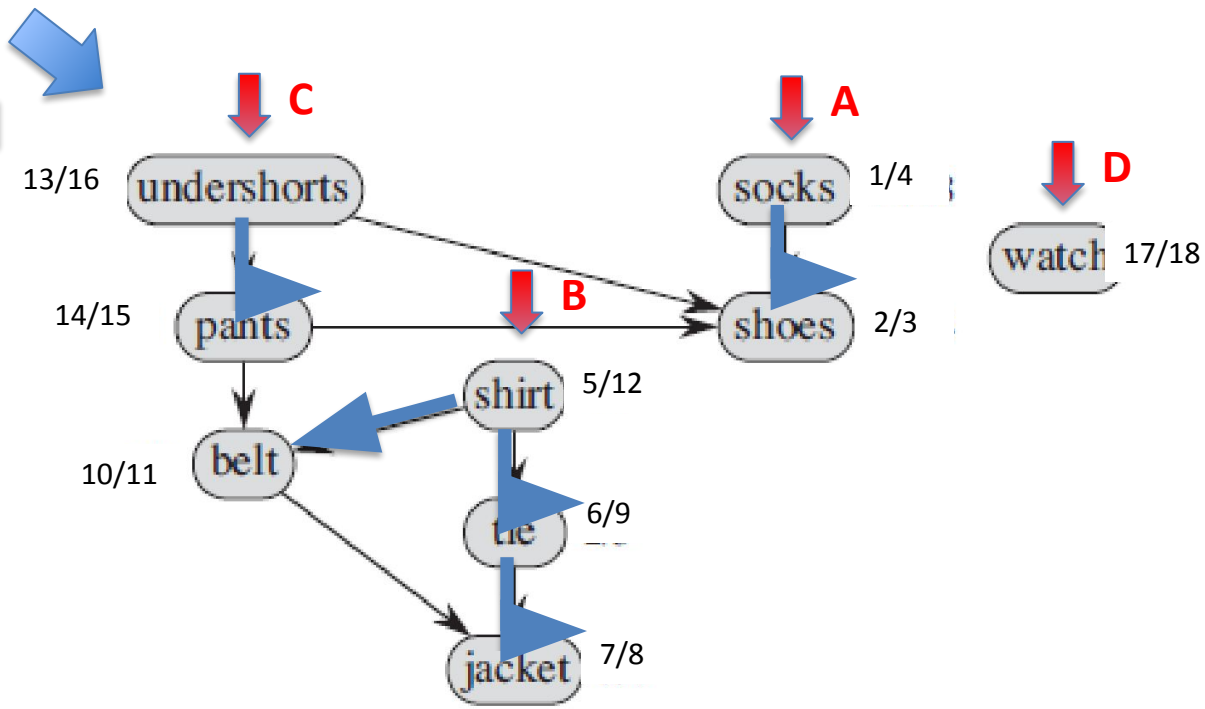
  - One sequence in which garments may be worn without violting the "pre-requisite" requirements



Depth-First Search

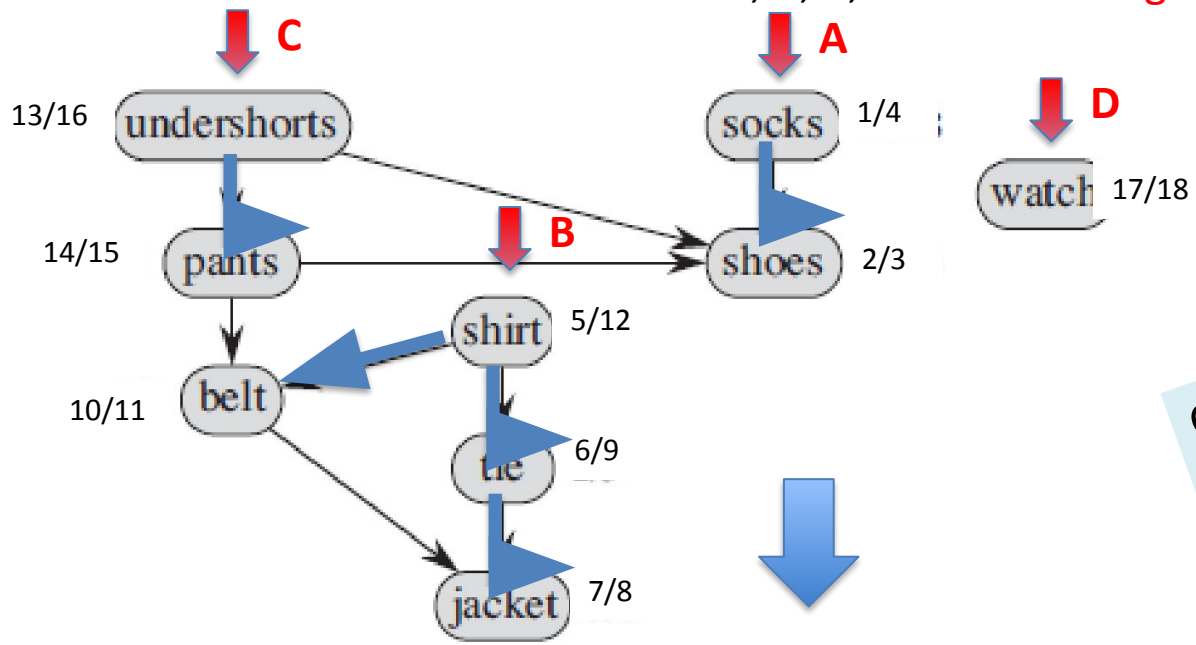
- Consider running DFS on graph below
  - DFS-Visit is run from vertices A, B, C, D in that changed order





Depth-First Search

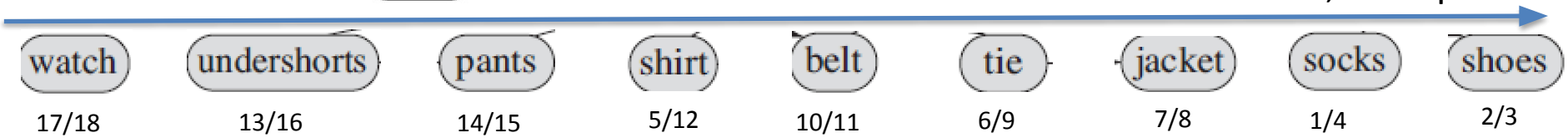
- Consider running DFS on graph below
  - DFS-Visit is run from vertices A, B, C, D in that changed order



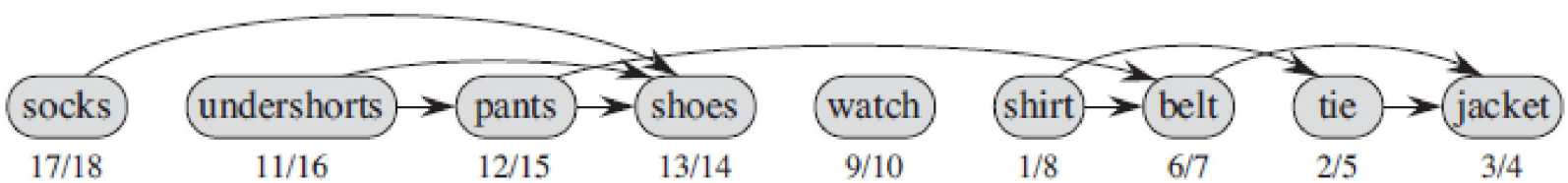
Choice of sequence of vertices from which DFS-Visit was run was arbitrary

⍰ Topological sort is NOT unique

Time, or sequence

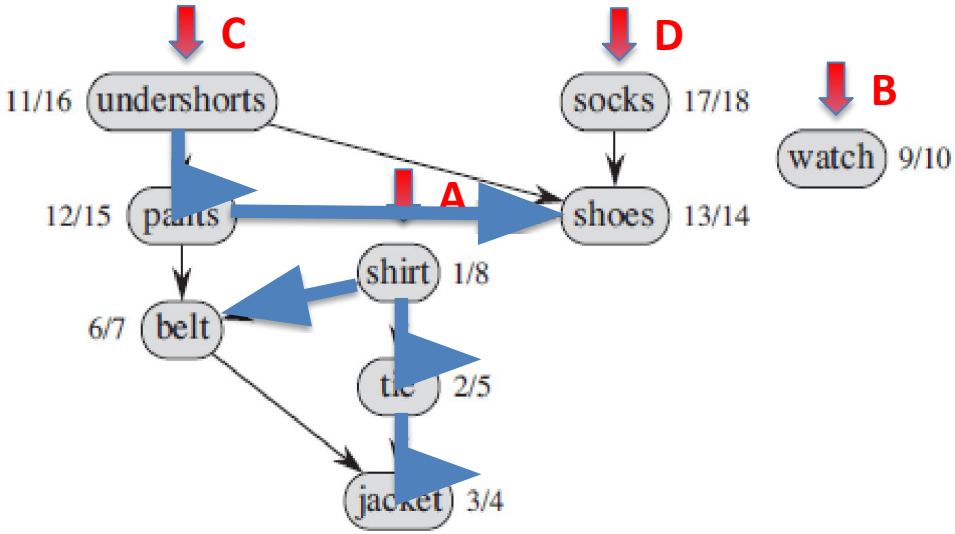


- Earlier:



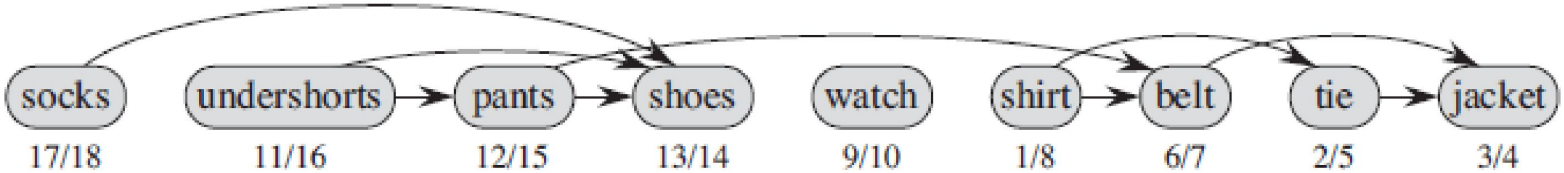
DFS, and topological-sort

- DFS when run results in the DFS forest, with “finish” time for each vertex



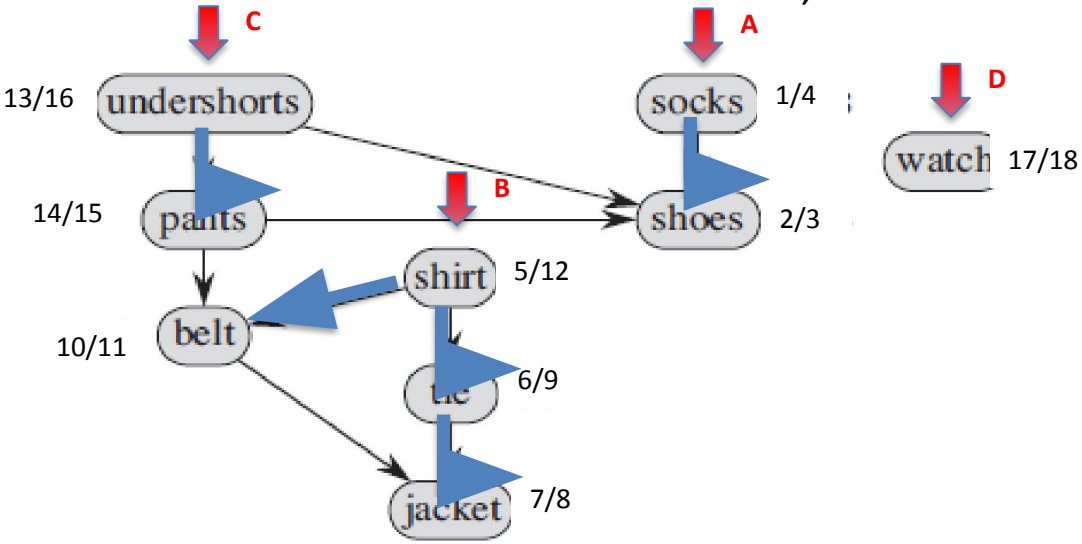
- To obtain a topological sort simply run the following algorithm:

```
TOPOLOGICAL-SORT(G)
call DFS(G) to compute finishing times v.f for each vertex v
as each vertex is finished, insert it onto the front of a linked list
return the linked list of vertices
```



DFS, and topological-sort

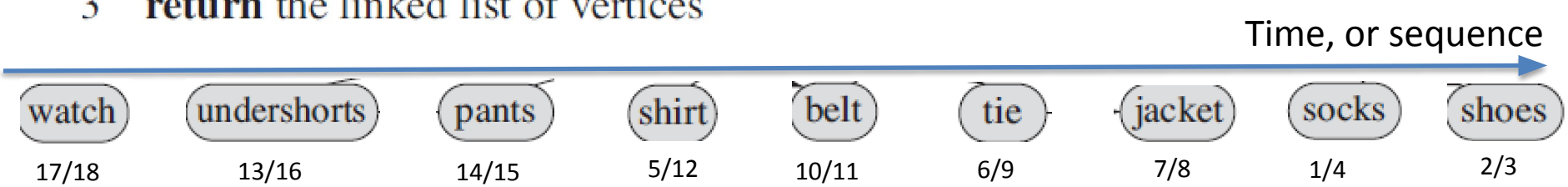
- DFS when run results in the DFS forest, with “finish” time for each vertex



- To obtain a topological sort simply run the following algorithm:

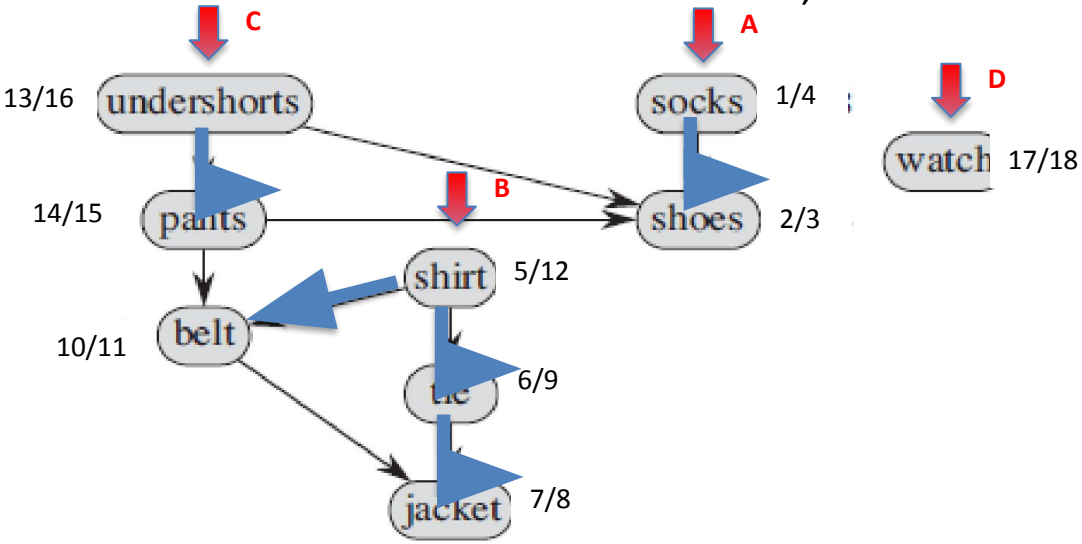
TOPOLOGICAL-SORT(G)

call DFS(G) to compute finishing times $v.f$ for each vertex v
as each vertex is finished, insert it onto the front of a linked list
return the linked list of vertices



DFS, and topological-sort

- DFS when run results in the DFS forest, with "finish" time for each vertex



The directed graph must be an "acyclic graph" if topological sort is to succeed

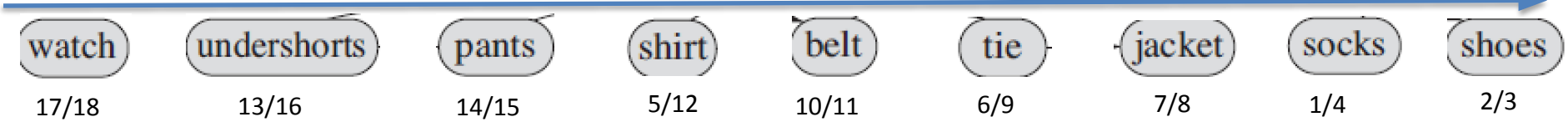
- To obtain a topological sort simply run the following algo

Any linear ordering of vertices in which all the arrows go to the right is a valid solution

TOPOLOGICAL-SORT(G)

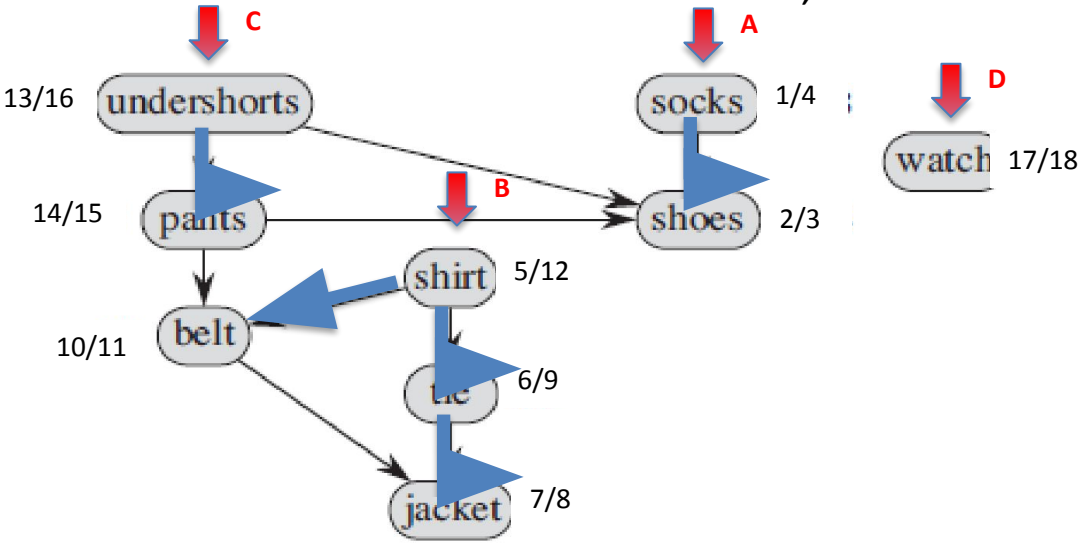
1 call DFS(G) to compute finishing times $v.f$ for
2 as each vertex is finished, insert it onto the front of a linked list
3 **return** the linked list of vertices

Time, or sequence



DFS, and topological-sort

- DFS when run results in the DFS forest, with "finish" time for each vertex



- To obtain a topological sort simply run the following algori

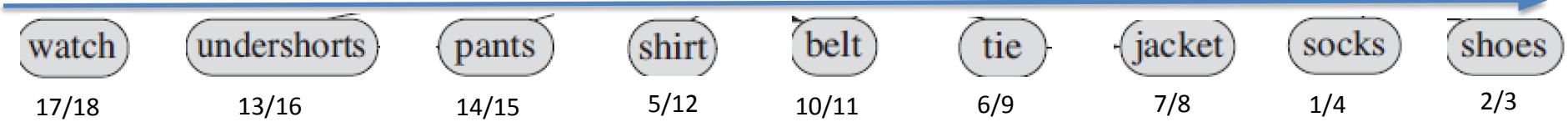
TOPOLOGICAL-SORT(G)

1 call DFS(G) to compute finishing times $v.f$ for e
2 as each vertex is finished, insert it onto the front of
3 **return** the linked list of vertices

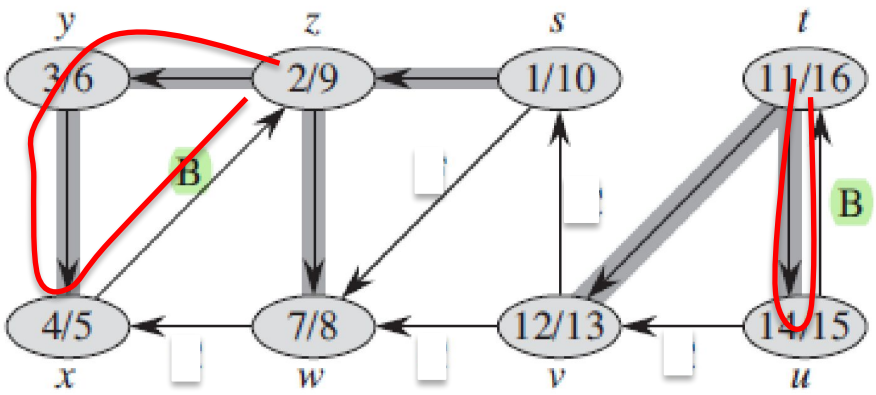
The directed graph must be an "acyclic graph" if topological sort is to succeed

The directed graph is acyclic if and only if while doing a DFS-Visit from a vertex one does not run into a "GRE vertex

Time, or sequence



DFS, and topological-sort



The directed graph must be an "acyclic graph" if topological sort is to succeed

The directed graph is acyclic if and only if a DFS does not yield a "BACK" edge. Equivalently, while discovery a vertex one does not run into a "GRE vertex

Also note that every vertex has at least one incoming edge, or in-degree(u) > 0

Topological sort algorithm

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

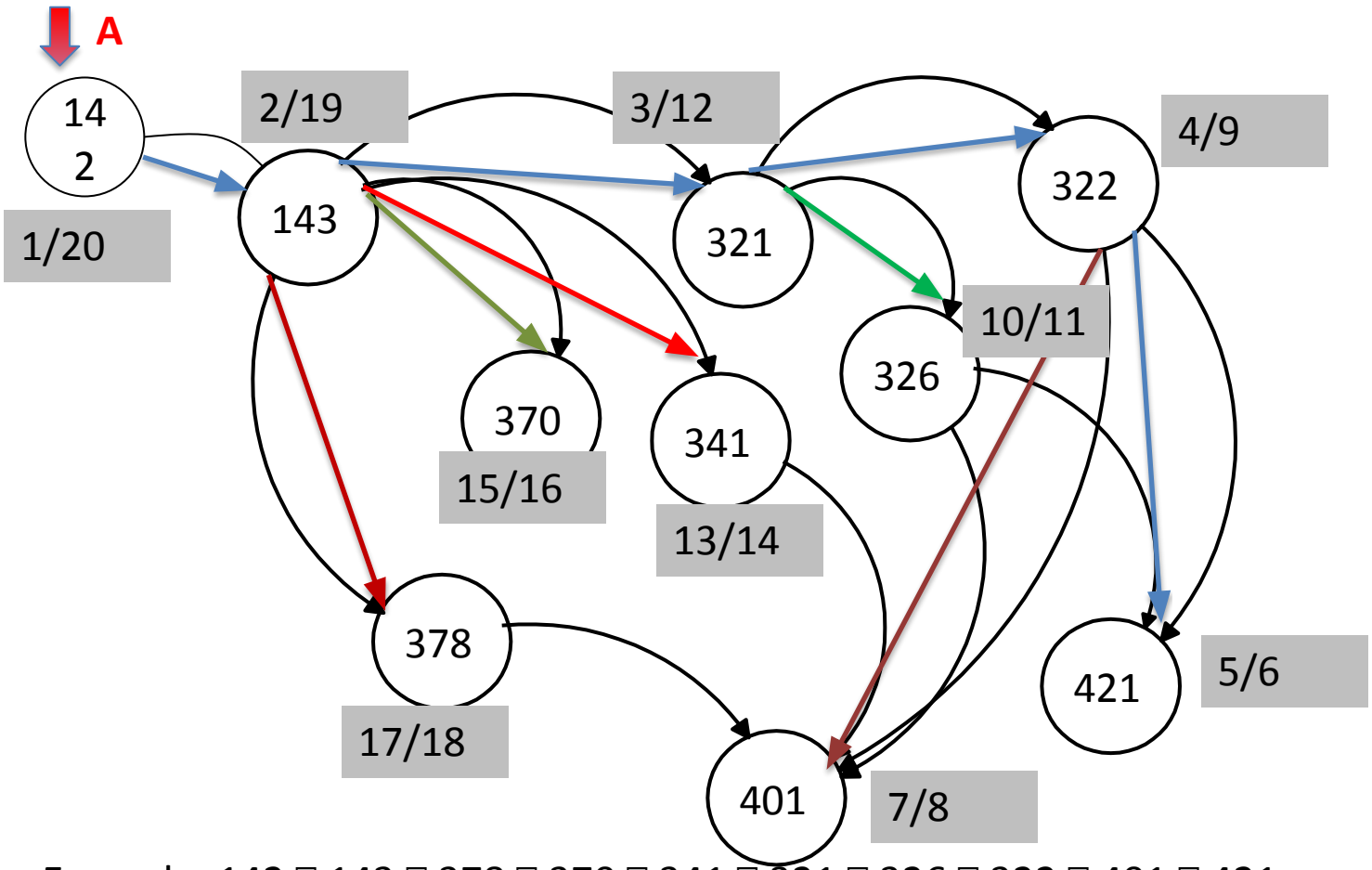
Init-List (L)

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

Add-First(L, u)

DFS, and topological-sort

- DFS when run results in the DFS forest, with “finish” time for each vertex



- Example: 142 ⇒ 143 ⇒ 378 ⇒ 370 ⇒ 341 ⇒ 321 ⇒ 326 ⇒ 322 ⇒ 401 ⇒ 421

Topological-sort: another algorithm

- DFS when run results in the DFS forest, with “finish” time for each vertex
- Step 0: Initialize list L = []
- Step 1: Identify a vertex u with no incoming edge
- Step 2: Delete vertex u from graph and all its outgoing edges from the graph. Add-Last(L, u)

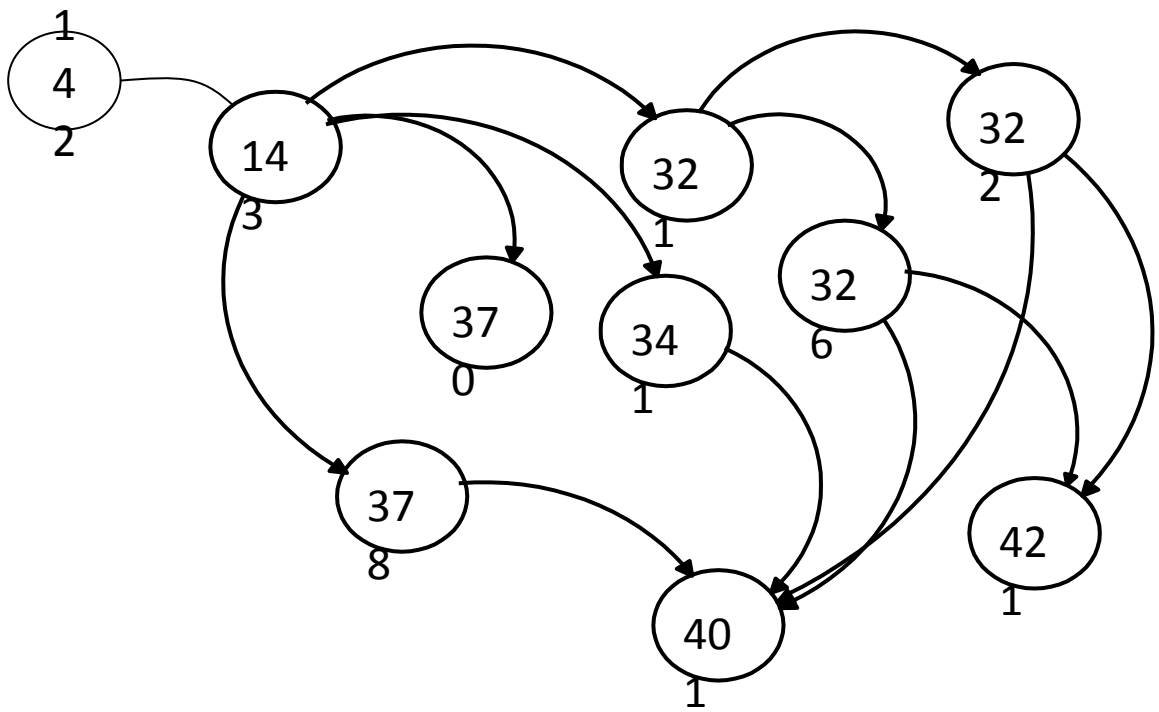
  Repeat steps 1 and 2, till graph is empty

If no such vertex, graph has a cycle ?
topological sort on graph that is not a DAG is NOT possible

Topological-sort: another algorithm

- DFS when run results in the DFS forest, with “finish” time for each vertex
- Step 0: Initialize list L = []
- Step 1: Identify a vertex u with no incoming edge
- Step 2: Delete vertex u from graph and all its outgoing edges from the graph. Add-Last(L, u)

Repeat steps 1 and 2, till graph is empty, Output L

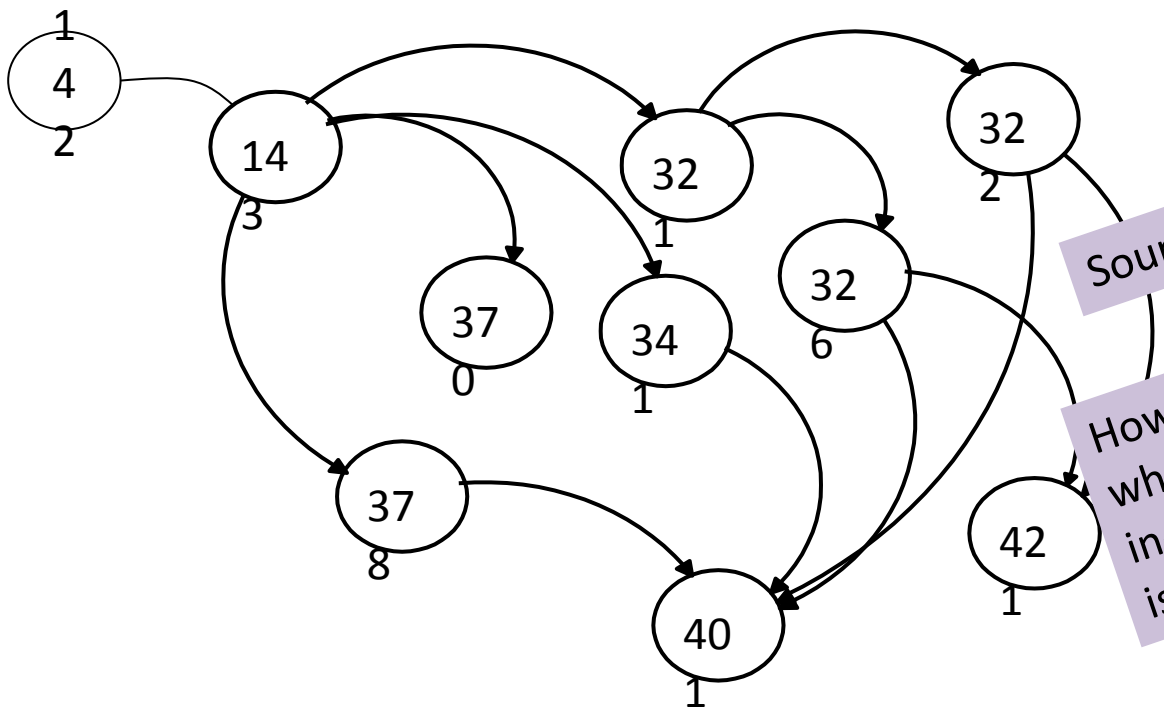


- Output list L = [142, 143, 378, 370, 341, 321, 326, 322, 401, 421]

Topological-sort: another algorithm

- DFS when run results in the DFS forest, with “finish” time for each vertex
- Step 0: Initialize list L = []
- Step 1: Identify a vertex u with no incoming edge
- Step 2: Delete vertex u from graph and all its outgoing edges from the graph. Add-Last(L, u)

Repeat steps 1 and 2, till graph is empty, Output L



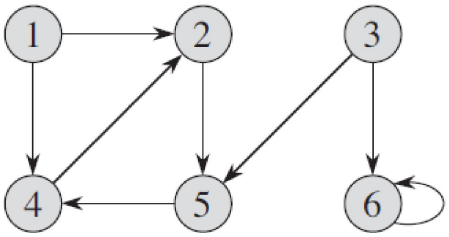
Sound very simple, elegant, BUT…

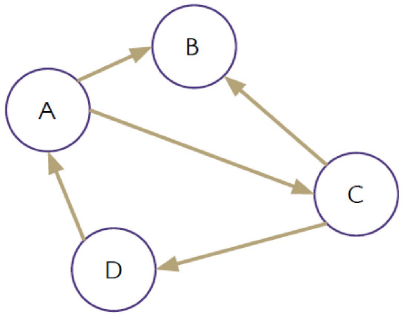
How does one determine as to whether a given vertex has any in-coming edge, particularly if a graph is represented using adjacency lists

- Output list L = [142, 143, 378, 370, 341, 321, 326, 322, 401, 421]

Try these examples

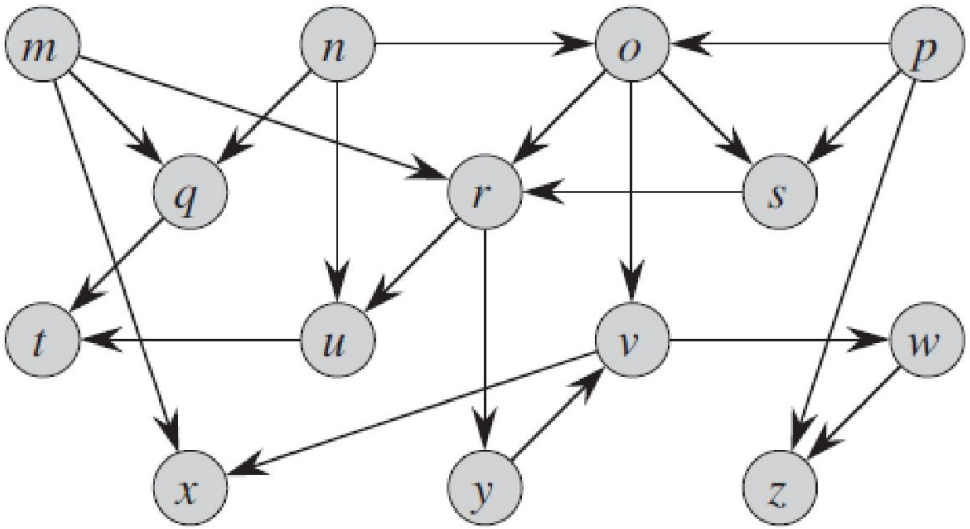
Can you topologically sort these graphs





Try these examples

Topologically sort this graph using both the algorithms



Time complexity of Topological sort

- Let G = (V, E) be an directed acyclic graph, and |V| = n, |E| = m.
- Then time complexity of Topological sot = time complexity of DFS, viz. O(n + m)

Q&A