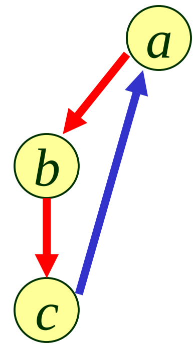
Applications of DFS

In $\boldsymbol{O}(|\boldsymbol{V}| + |\boldsymbol{E}|)$ time, we can

- Find connected components of G.
- Determine if G has a cycle.
- Determine if removing a vertex or edge will disconnect G.
- Determine if G is *planar*, i.e., if it can be drawn on the plane with no crossing edges.
- …

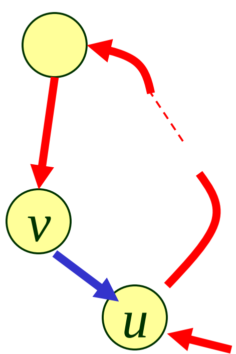
Detecting Cycles

Fact A directed graph G has a cycle if and only if its DFS forest has a **back edge**.



$\Leftarrow$ A back edge leads to a cycle (adding an edge to a tree).

$\Rightarrow$ Suppose there is a cycle. Let u be the first vertex discovered on the cycle and v be the vertex such that the edge $\langle v, u \rangle$ is in the cycle.



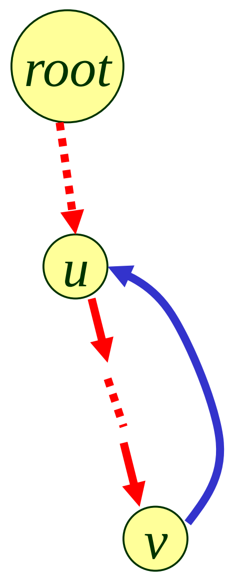
- v has not been explored at the time of the initial call to dfsVisit(u).
- v will be visited before returning from dfsVisit(u).

Therefore at the time of visiting v, a back edge $\langle v, u \rangle$ will be found.

The fact also holds for an undirected graph.

Using DFS

A back edge can be easily detected during DFS.

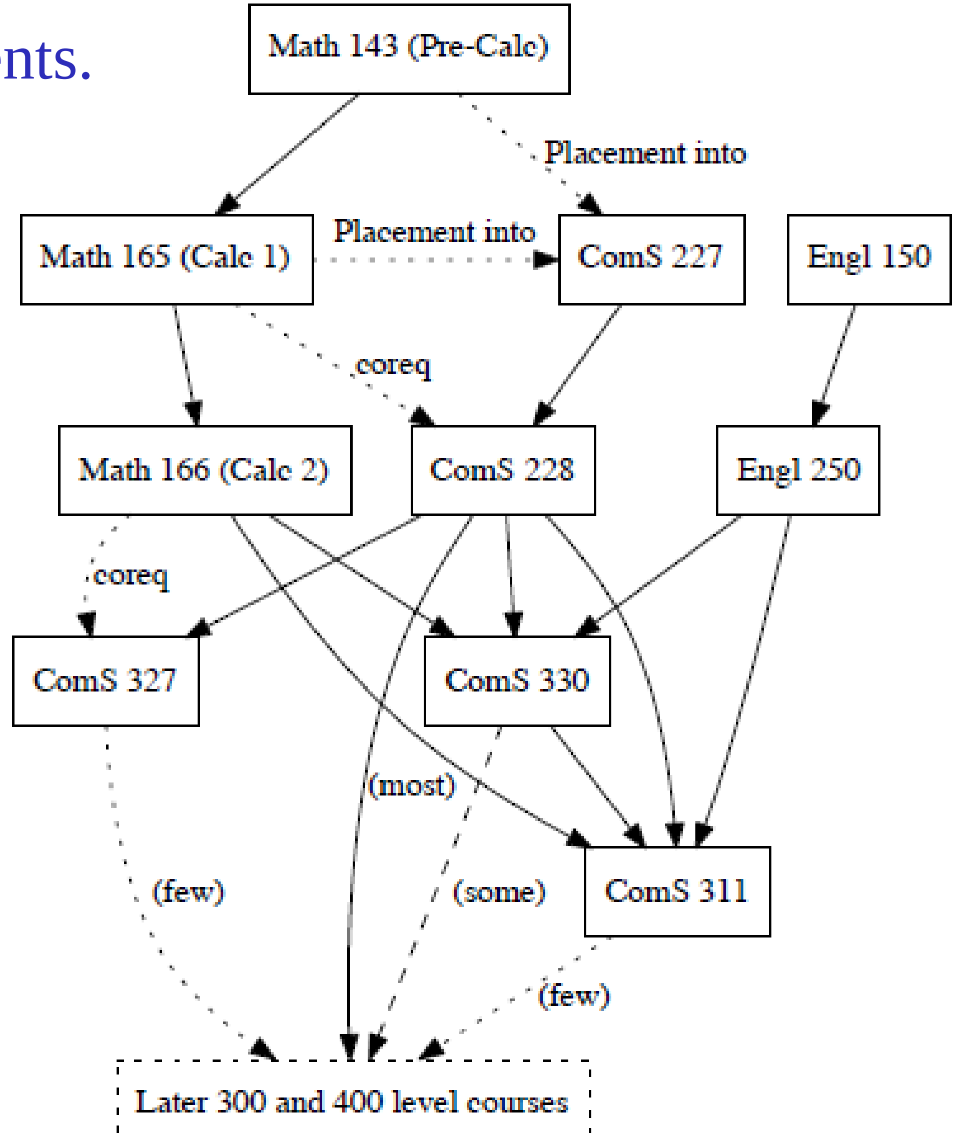


We can test if a graph is acyclic in $O(|V| + |E|)$ time.

Directed Acyclic Graph (DAG)

Model precedences among events.

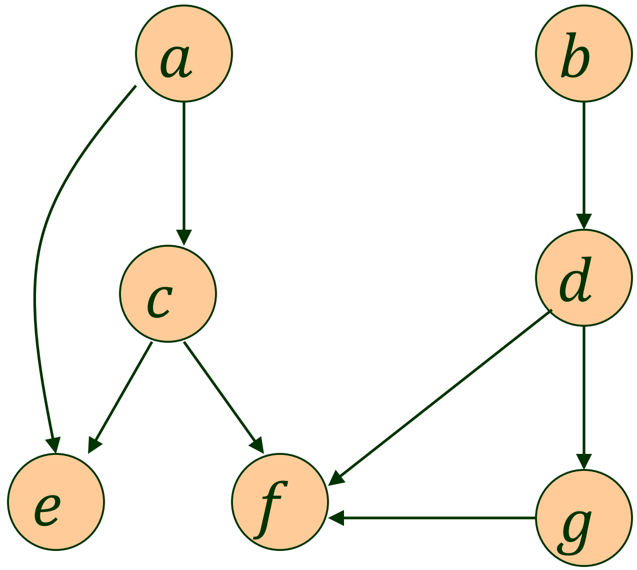
Com S course flowchart:



How to plan courses?

Topological Sort of DAGs

Ordering $<$ over vertices: $u < v$ whenever $\langle u, v \rangle$ is an edge.



Some topological sorts:

a, c, e, b, d, g, f
a, b, c, d, g, f, e
b, d, g, a, c, f, e

How about $b, a, d, c, e, \underline{f, g}$?

order violation

$a < c, \quad a < e, \quad c < e, \quad c < f$
$b < d, \quad d < g, \quad d < f, \quad g < f$

Intuition: Precedence Diagram

- Each node represents an activity; e.g., taking a class.
- $\langle u, v \rangle \in E(G)$ implies activity u must be scheduled before activity v.
- Topological sort schedules all activities.
- More than one schedule may exist.

Topological Sort Algorithm

Fact: G can be topologically sorted if and only if it has no cycle, that is, if and only if it is a DAG.

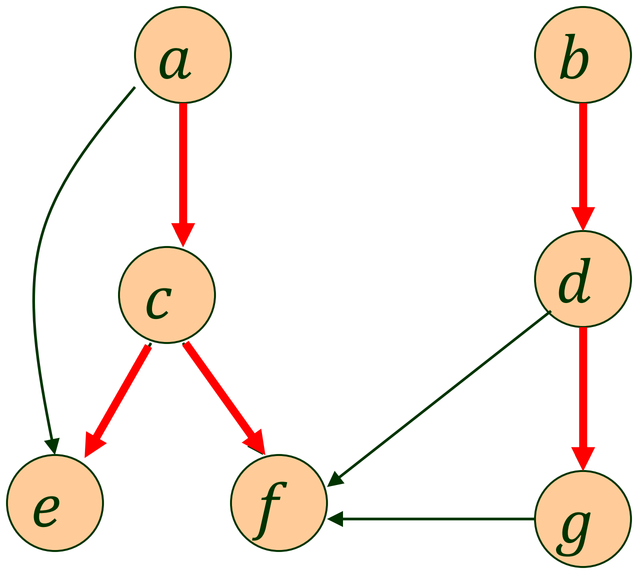
TOPOLOGICALSORT(G)

1. call dfs(G)

2. when a node v is finished, insert it onto the front of a linked list L

3. return L

Execution



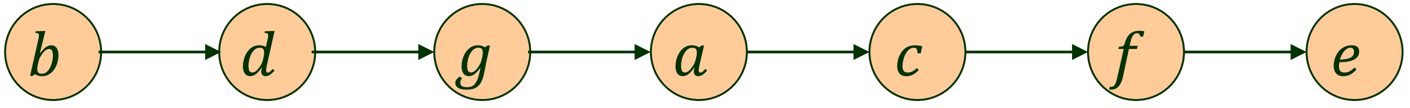
TOPOLOGICALSORT(G)

1. call dfs(G)

2. when a node v is finished, insert it onto the front of a linked list L

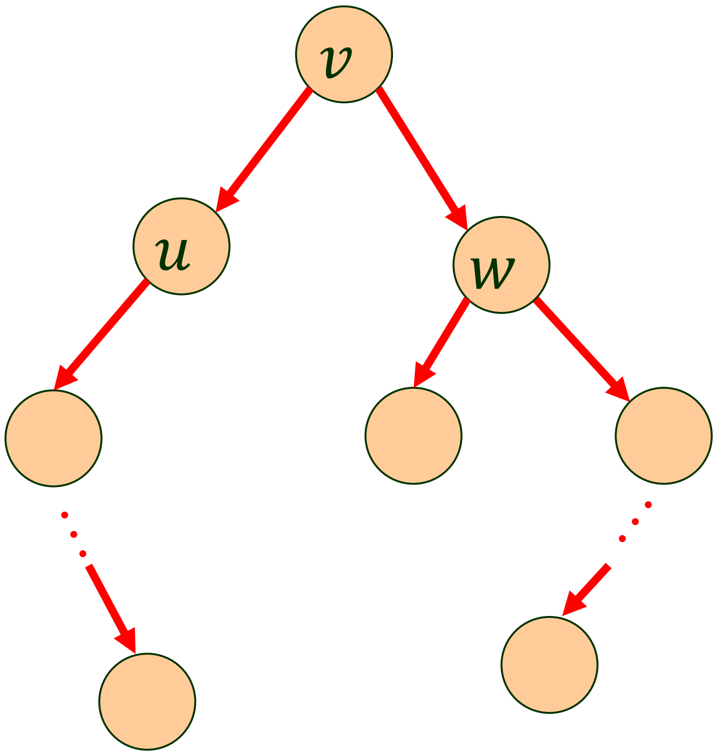
3. return L

L (with the nodes in a topological order):



Why Correct?

At the moment a node v is finished:



DFS subtree rooted at v