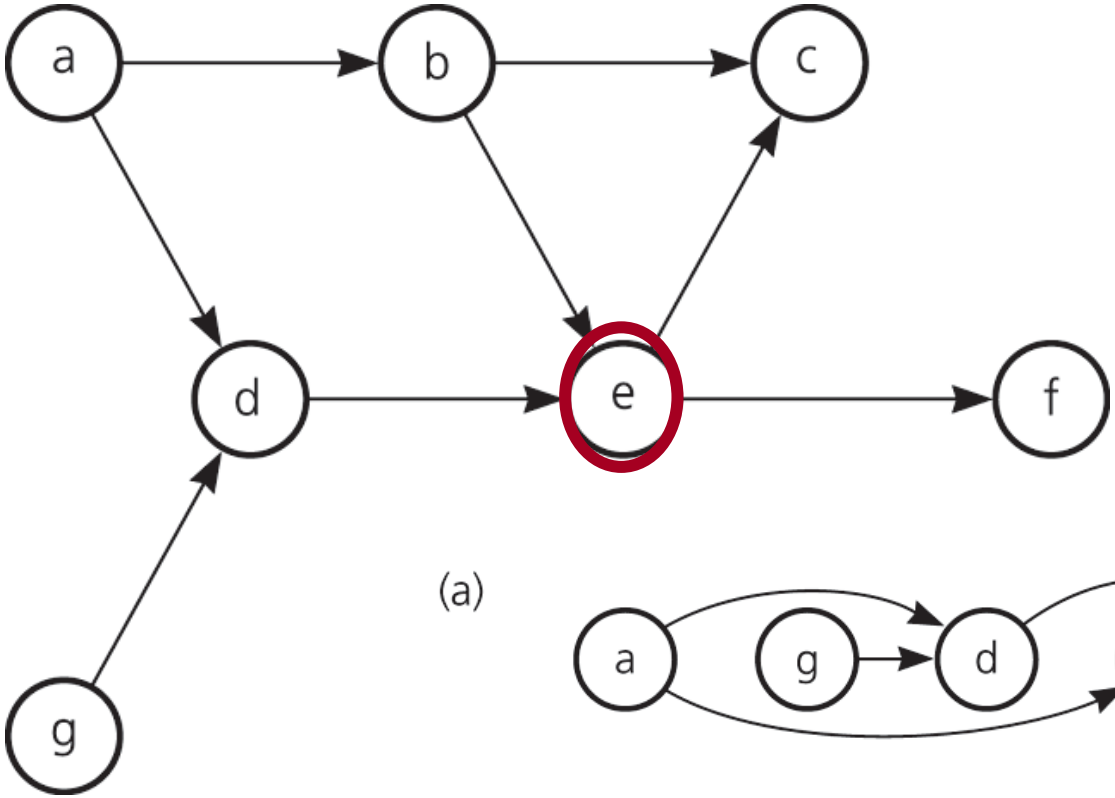
Graph App.

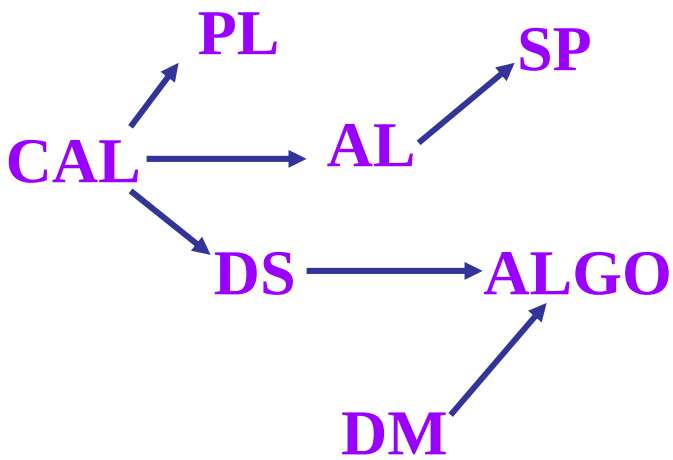
- Topological Sort
- Spanning Tree
- Shortest Paths

Topological Sort: *Definition*

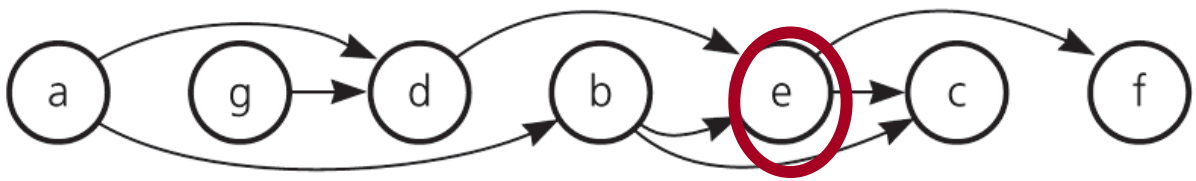
- **Topological order**
  - A list of vertices in a directed graph without cycles (Acyclic Digraph or Directed Acyclic Graph, DAG) such that vertex *x* precedes vertex *y* if there is a directed edge from *x* to *y* in the graph
  - Several topological orders are possible for a given graph
- **Topological sorting**
  - Arranging the vertices into a *topological order*

Topological Sort: *Examples*

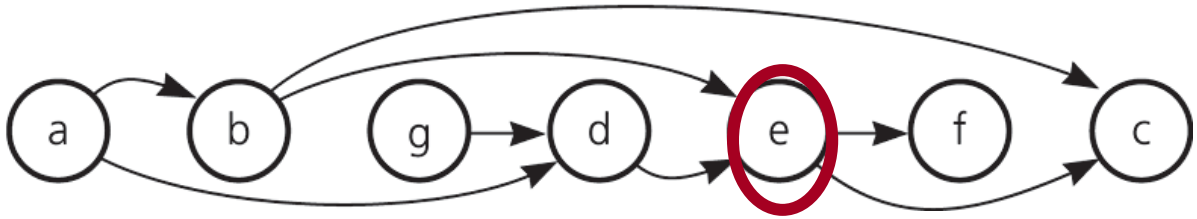




(a)



(b)

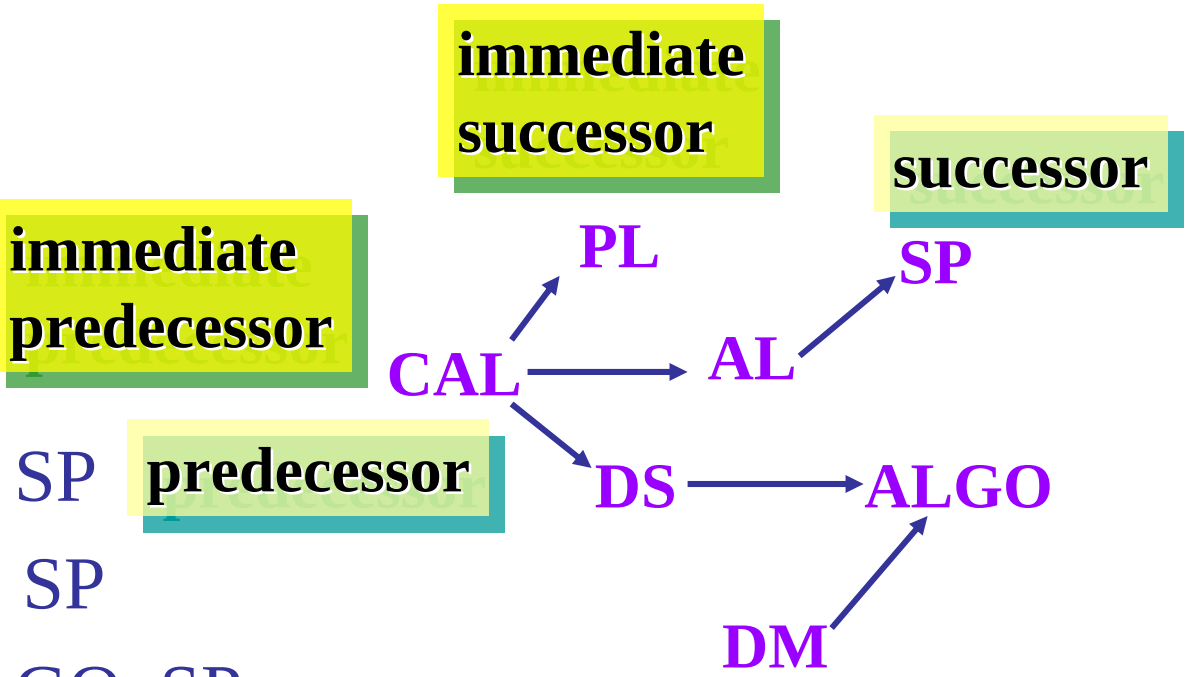


Topological Sort: *Algorithms*

- ***topSort1***
  1. Find a vertex that has no successor (out-degree=0)
  2. Add the vertex to the *beginning* of a list
  3. Remove that vertex from the graph, as well as all edges that lead to it
  4. Repeat the previous steps until the graph is *empty*
     - **When the loop ends, the list of vertices will be in *topological order***

A Trace of *topSort1* (concept)

- **List**
  - SP
  - ALGO, SP
  - DM, ALGO, SP
  - PL, DM, ALGO, SP
  - DS, DM, ALGO, SP
  - AL, DS, DM, ALGO, SP
  - CAL, AL, DS, DM, ALGO, SP

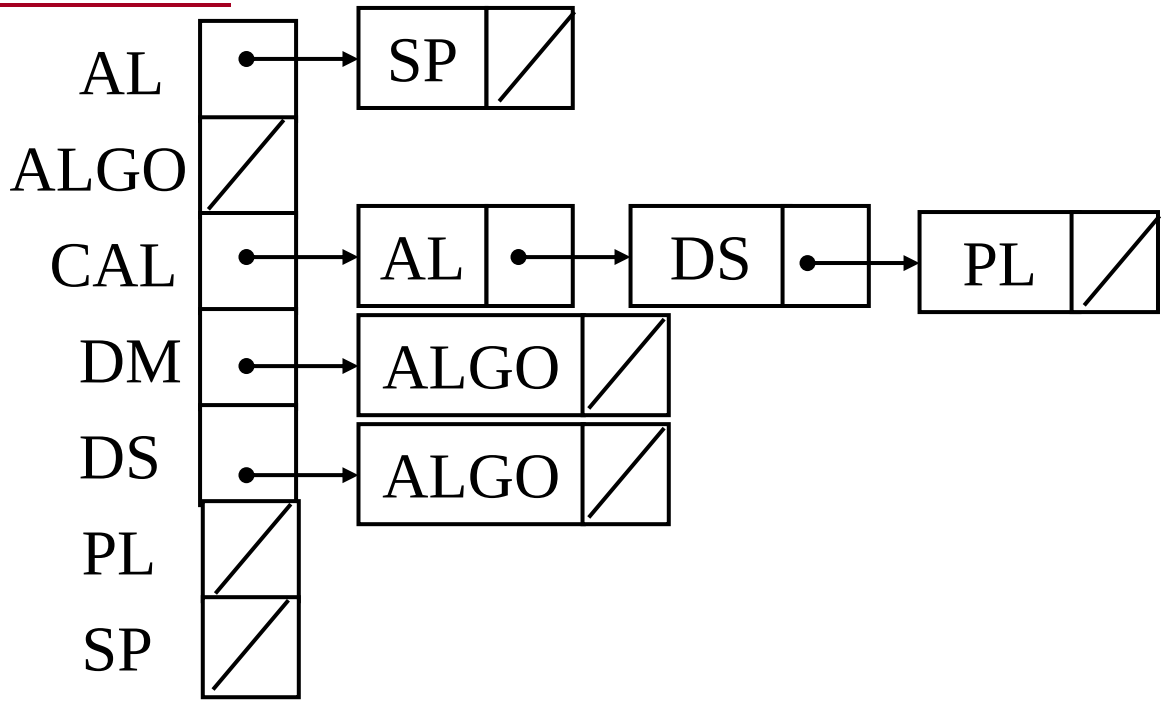


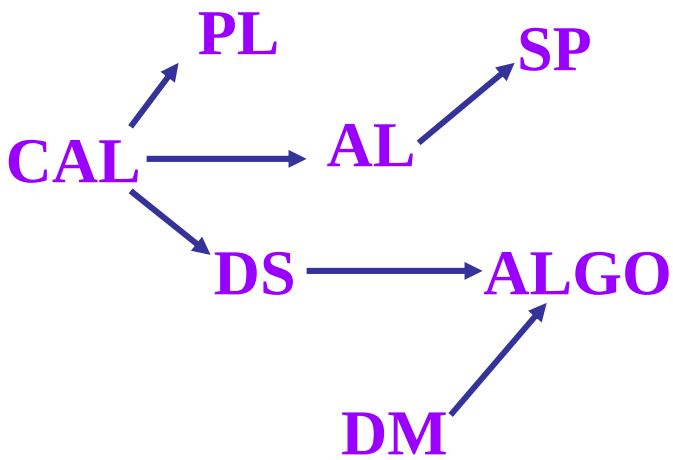
Practice 3: *in-degree* = 0 (no *predecessor*)

in-degree

Number of immediate predecessors

immediate successors





CAL

CAL, PL

CAL, PL, DM

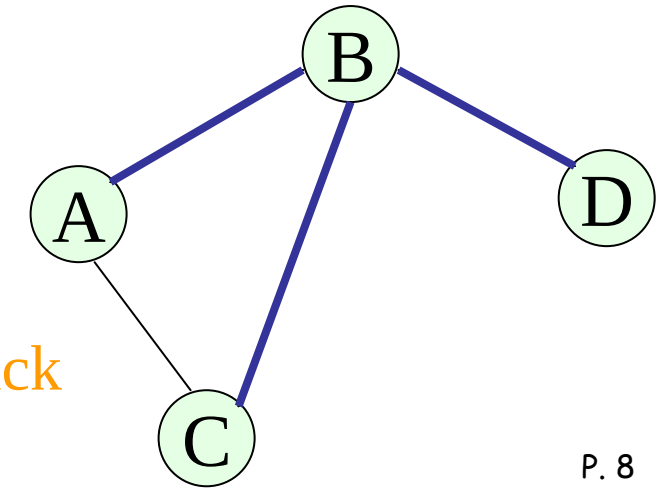
…

Topological Sort: *Algorithms*

- ***topSort2***
  - A modification of the *iterative DFS* algorithm
  - Push all vertices that have no predecessor onto a stack
  - Each time you pop a vertex from the stack, add it to the *beginning* of a list of vertices
  - When the traversal ends, the list of vertices will be in topological order

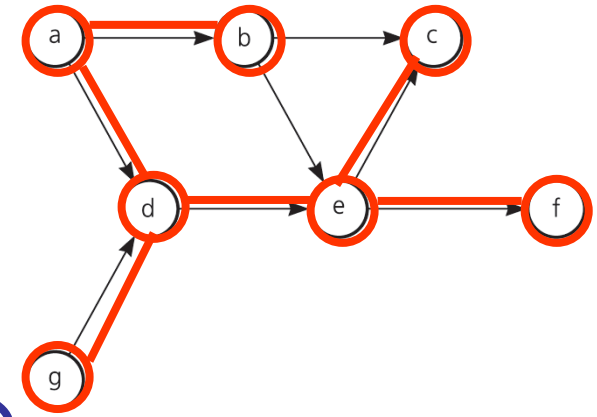
DFS in *iterative* form (stack)


```
iterativeDFS(Vertex v)
    s.createStack();
    s.push(v);
    Mark v as visited;
    while (!s.isEmpty())
    {     u = s.getTop();                    // at top of the stack
          if (unvisited vertex w is adjacent to u)
          {         s.push(w);
                    Mark w as visited;
          }
          else      s.pop();                   // backtrack
    }
```



A Trace of *topSort2*

Action	Stack s (bottom to top)	List aList (beginning to end)
Push a	a	
Push g	a g	
Push d	a g d	
Push e	a g d e	
Push c	a g d e c	c
Pop c, add c to aList	a g d e	c
Push f	a g d e f	f c
Pop f, add f to aList	a g d e	e f c
Pop e, add e to aList	a g d	d e f c
Pop d, add d to aList	a g	g d e f c
Pop g, add g to aList	a	g d e f c
Push b	a b	b g d e f c
Pop b, add b to aList	a	a b g d e f c
Pop a, add a to aList	(empty)	



Q&A: how to do it in the *reverse* order?

Self-exercise 3

1. Using the **topological sort** algorithm ***topSort1*** (*finding the vertex without successor first*), as given in classes, write the **topological order** of the vertices for each graph.

PS. *If you have multiple choices, find the vertex with the smallest label first.*