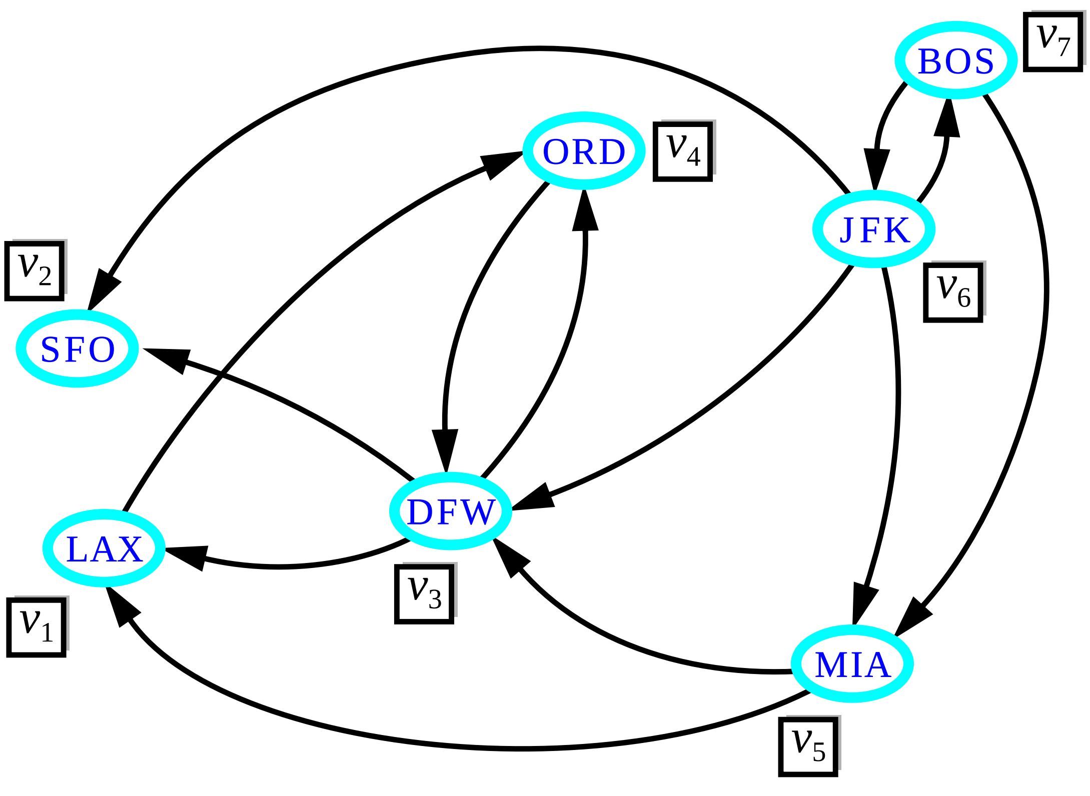
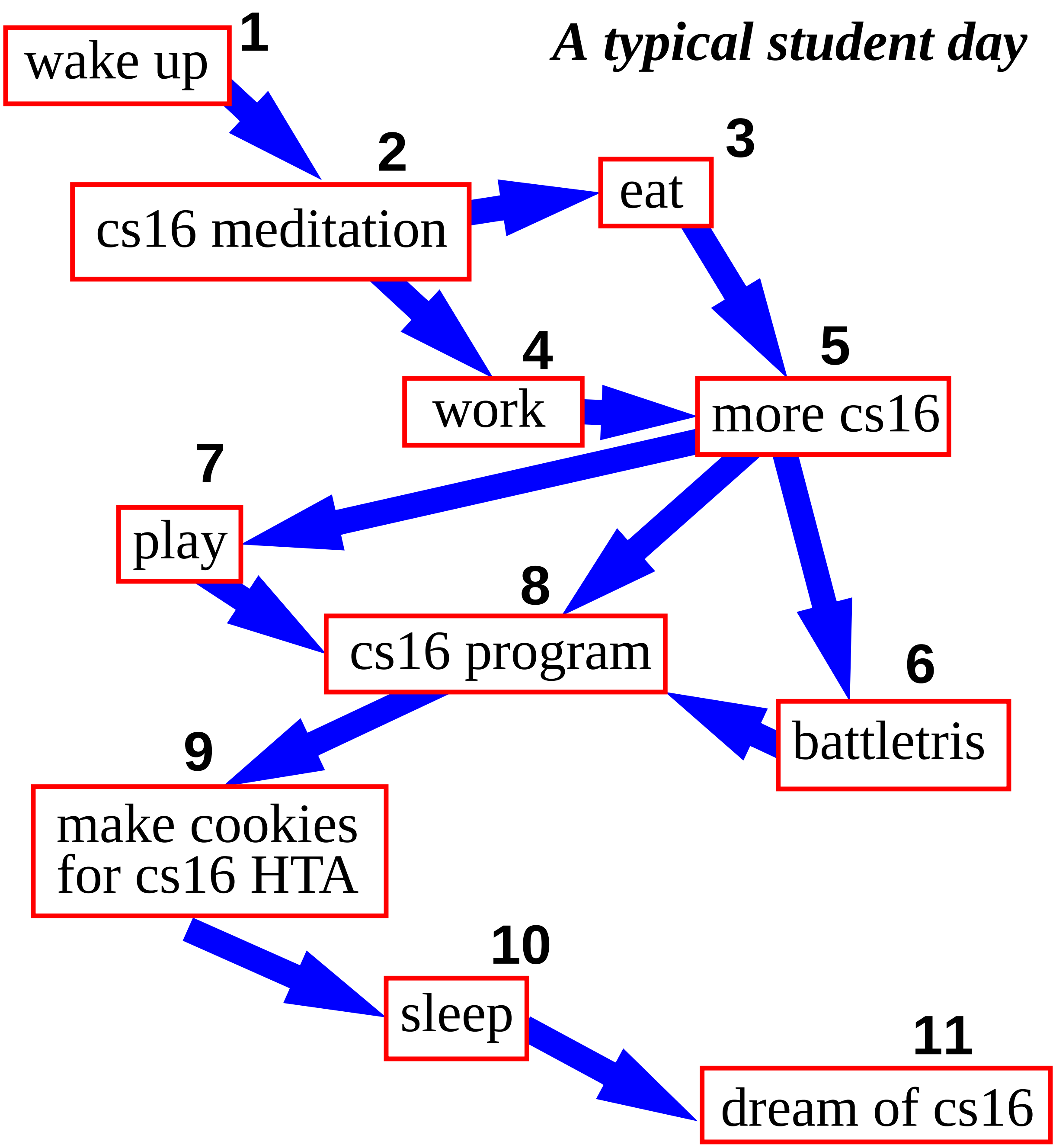
DIGRAPHS

- Reachability
- Connectivity
- Transitive Closure
- Floyd-Warshall Algorithm



DIGRAPHS

A typical student day



What’s a Digraph?

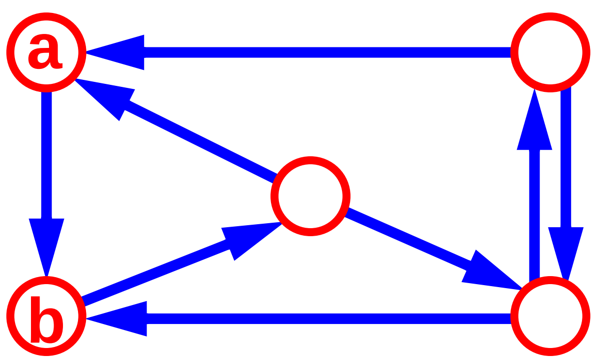
a) A small burrowing animal with long sharp teeth and a unquenchable lust for the blood of computer science majors

b) A distressed graph

c) A directed graph

Each edge goes in one direction

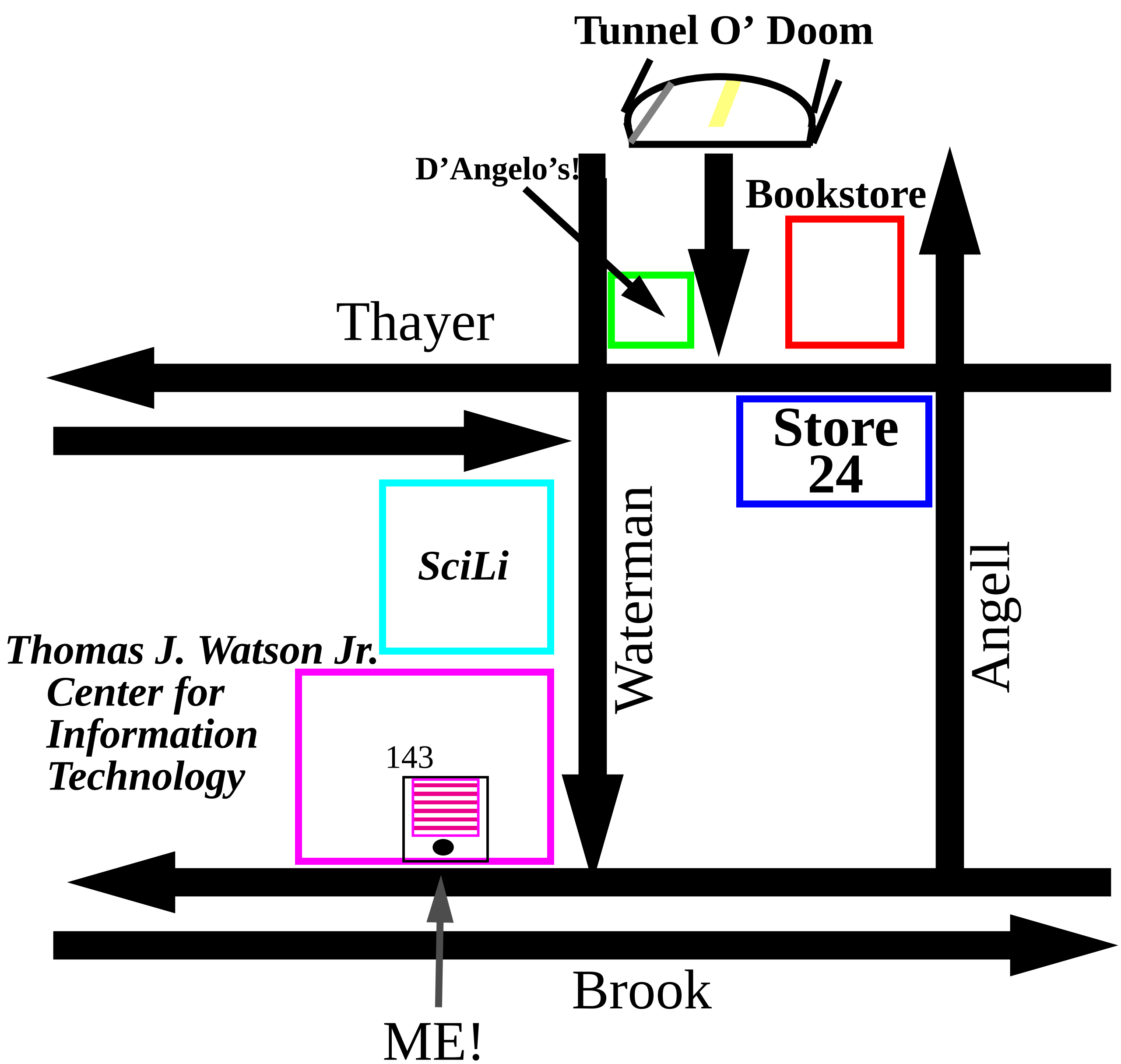
Edge (a,b) goes from a to b, but not b to a



You’re saying, “Yo, how about an example of how we might be enlightened by the use of digraphs!!” – Well, if you insist. . .

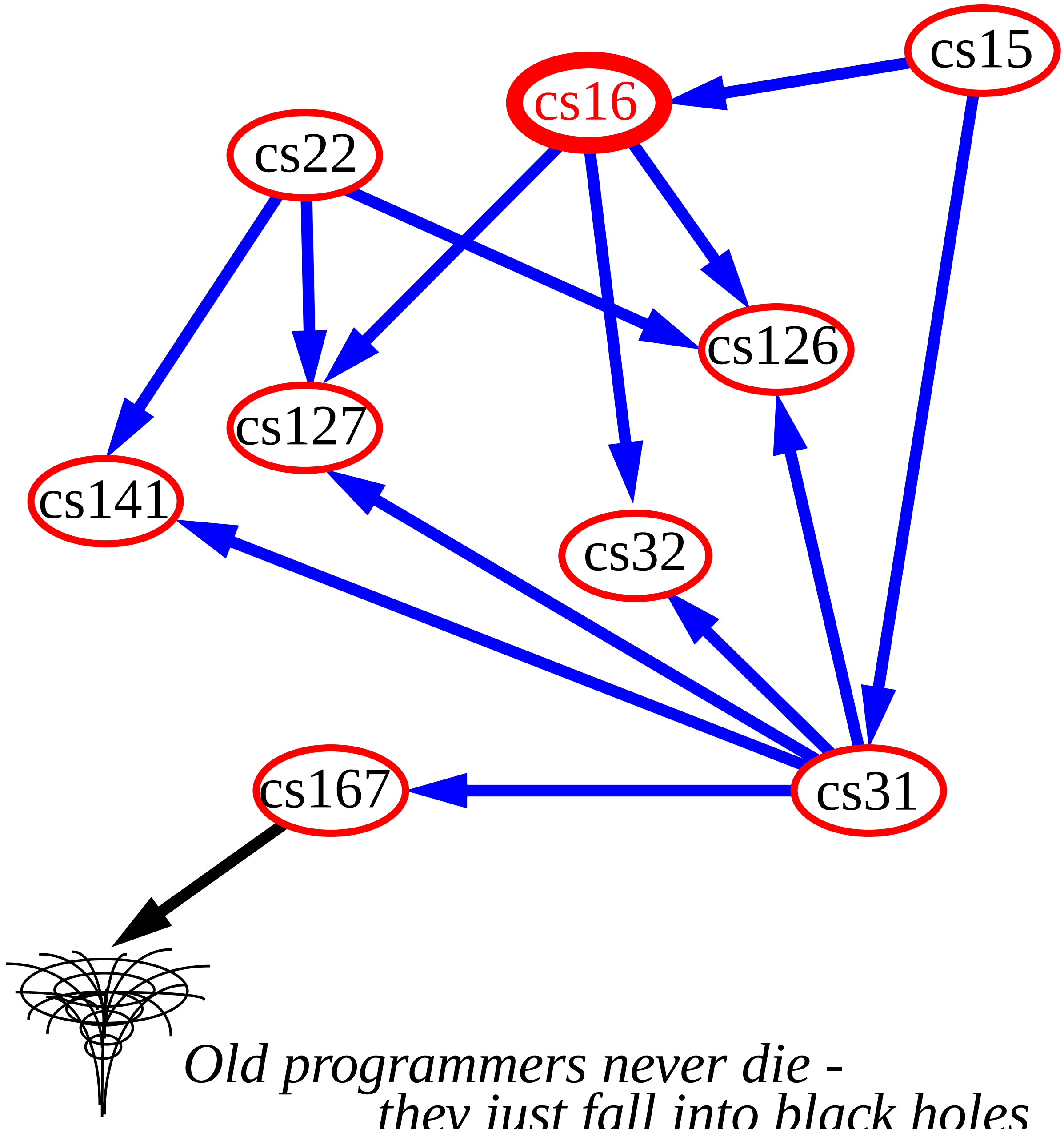
Applications

Maps: digraphs handle one-way streets
(especially helpful in Providence)



Another Application

Scheduling: edge (a,b) means task a must be completed before b can be started



Old programmers never die -
they just fall into black holes

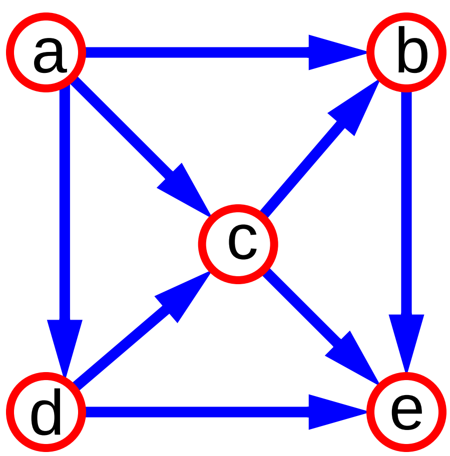
DAG's

dag: (noun) dÂ-g

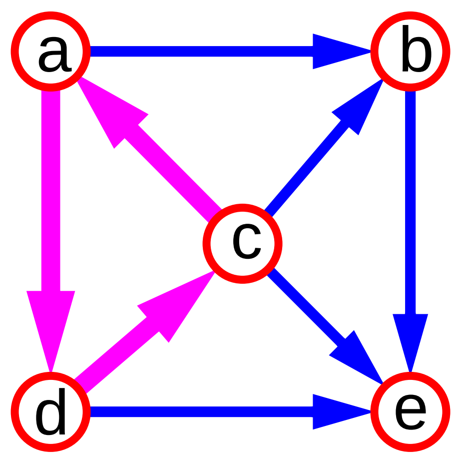
1. **D**i-**A**cyl-**G**lycerol – My favorite snack!
2. "~~man's~~ person's best friend"
3. **directed acyclic graph**

Say What?!

directed graph with no directed cycles



DAG

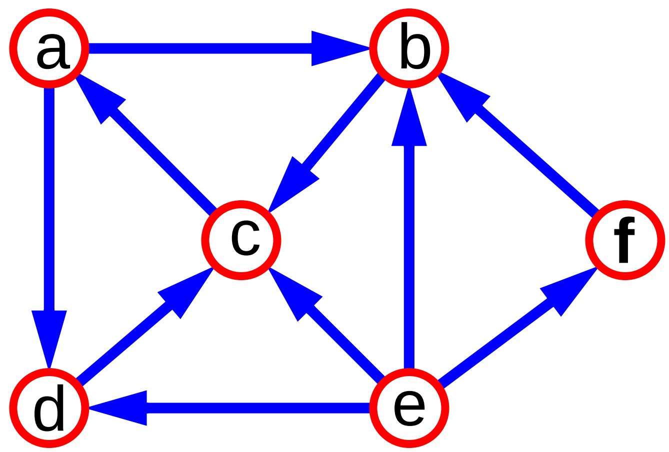


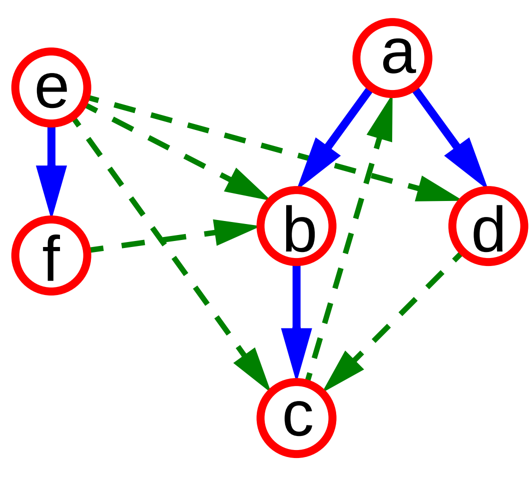
not a DAG

Depth-First Search

Same algorithm as for undirected graphs

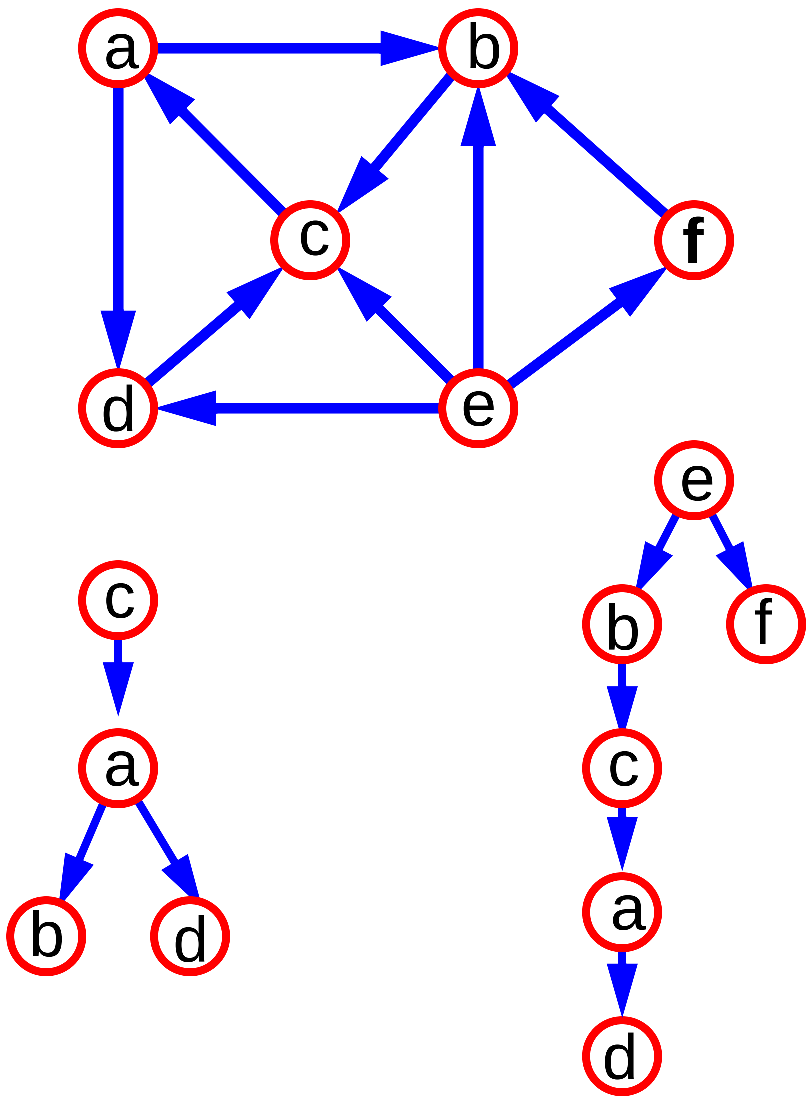
On a connected digraph, may yield unconnected DFS trees (i.e., a DFS forest)





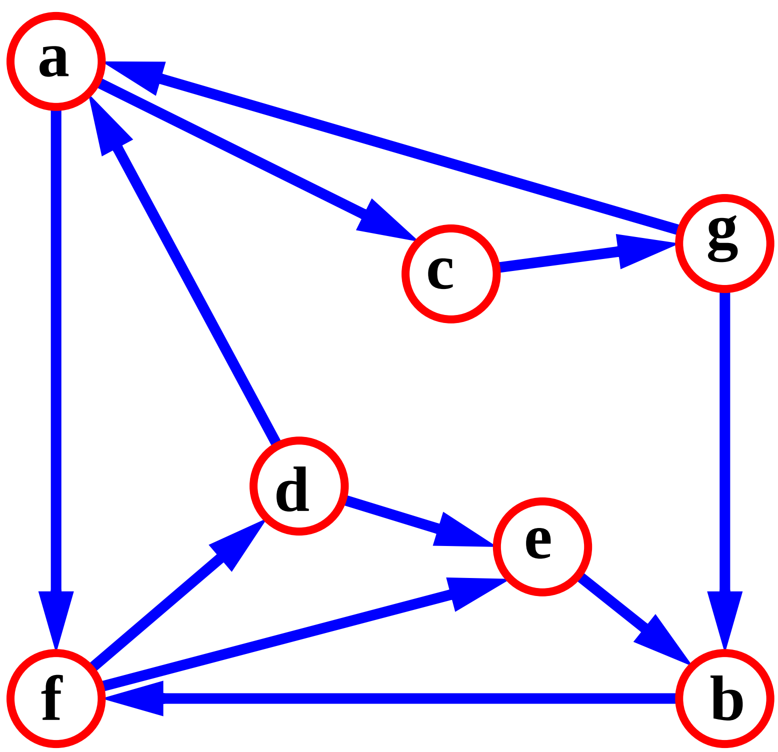
Reachability

DFS tree rooted at v: vertices reachable from v via directed paths

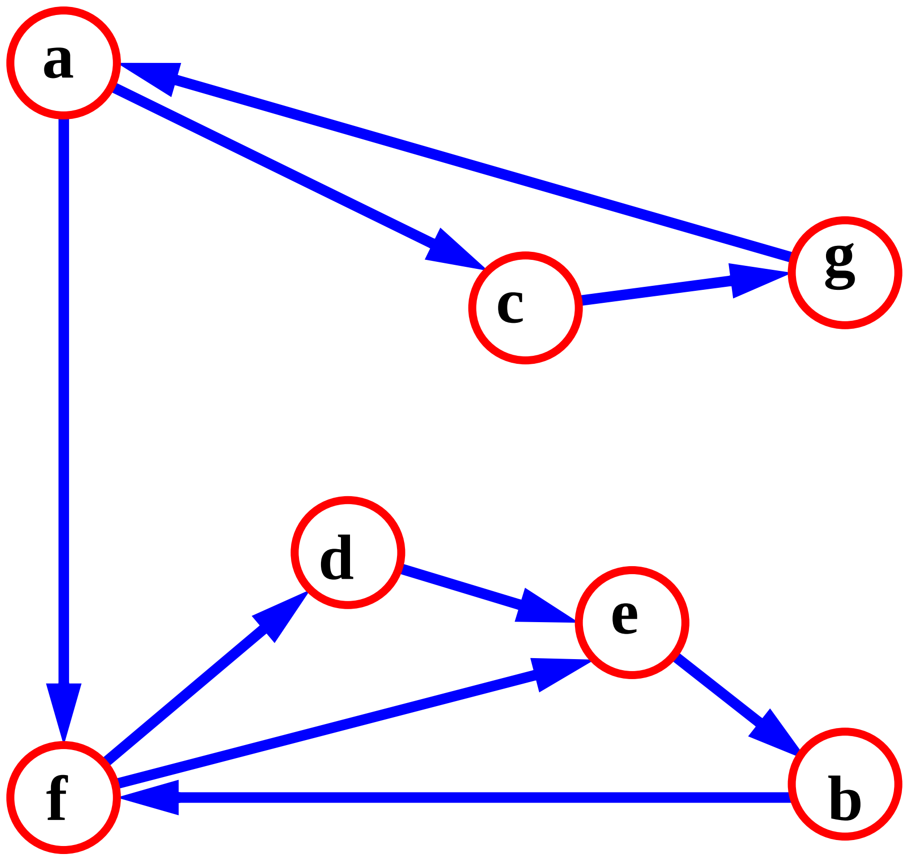


Strongly Connected Digraphs

Each vertex can reach all other vertices



Strongly Connected Components



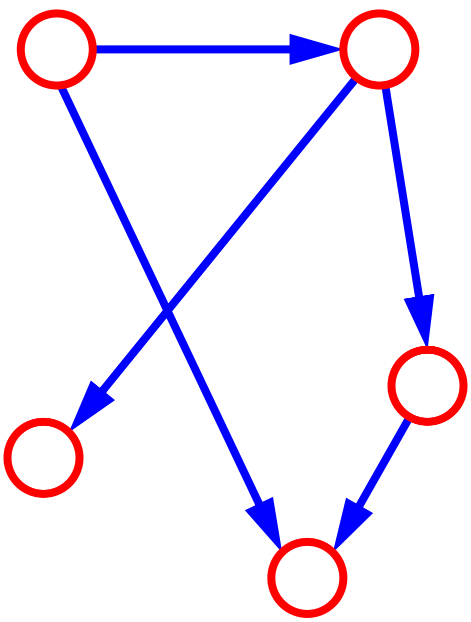
{ a , c , g }

{ f , d , e , b }

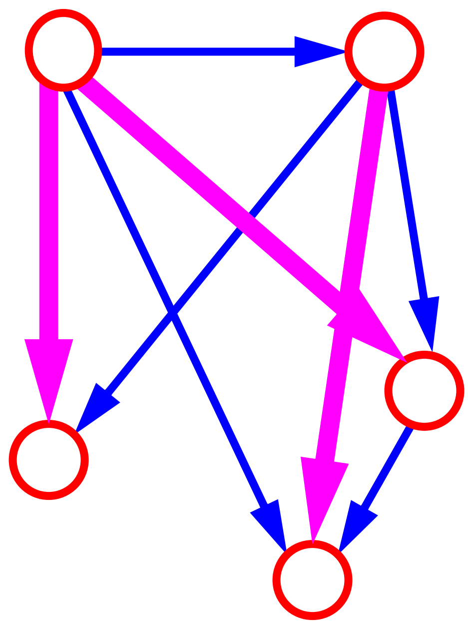
Transitive Closure

Digraph G* is obtained from G using the rule:

If there is a directed path in **G** from a to b, then add the edge (a,b) to **G***



G



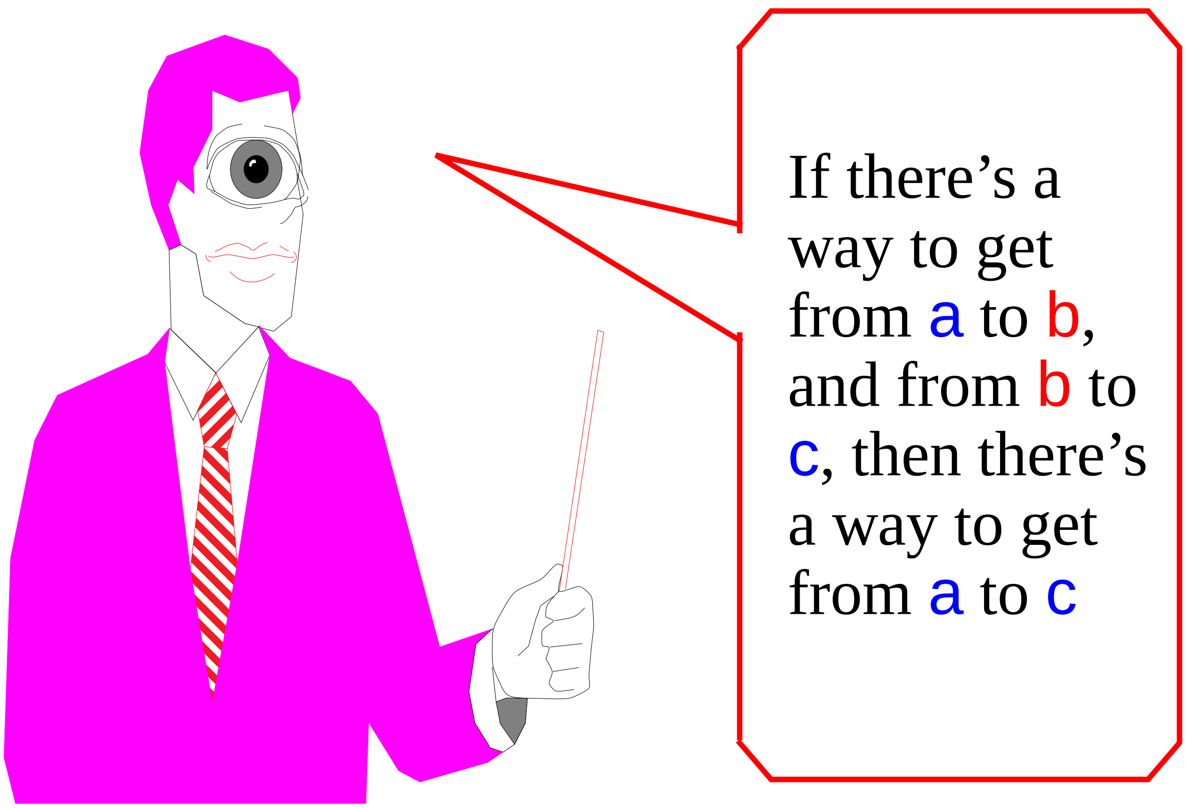
G*

Computing the Transitive Closure

We can perform DFS starting at each vertex

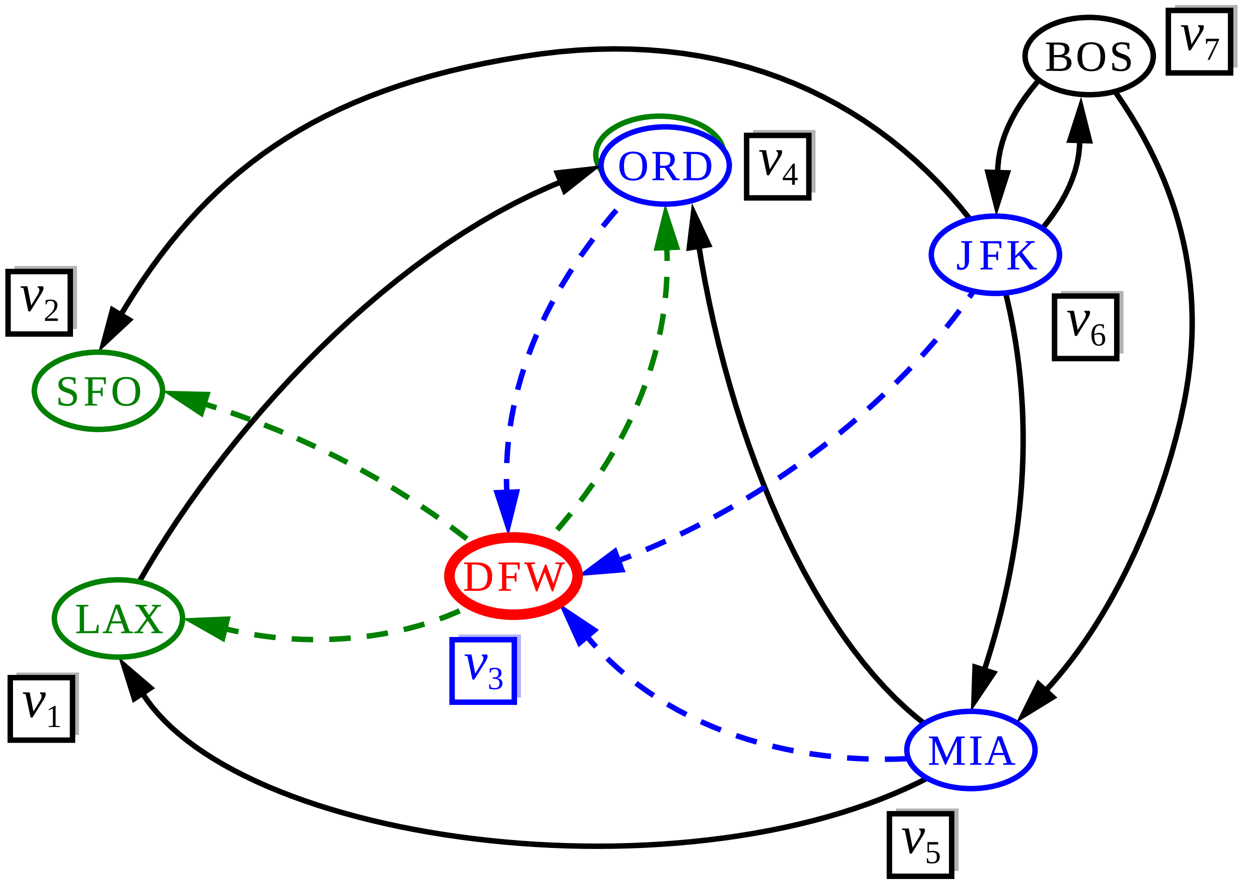
Time: **O(n(n+m))**

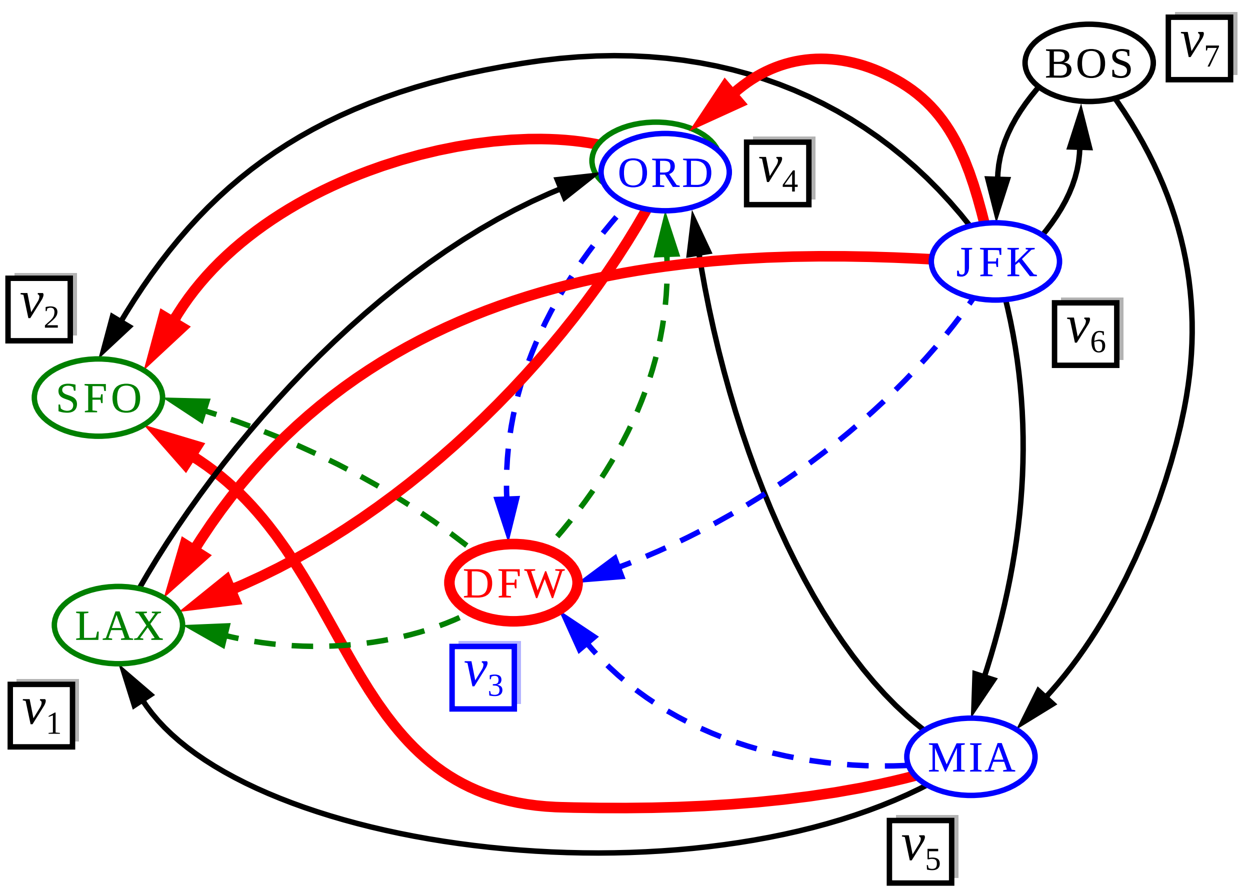
Alternatively ... Floyd-Warshall Algorithm:



"Pink" Floyd

Example


BOS
v_7
ORD
v_4
JFK
v_6
v_2
SFO
DFW
LAX
v_3
v_1
MIA
v_5


BOS
v_7
ORD
v_4
JFK
v_6
v_2
SFO
DFW
LAX
v_3
v_1
MIA
v_5

Floyd-Warshall Algorithm

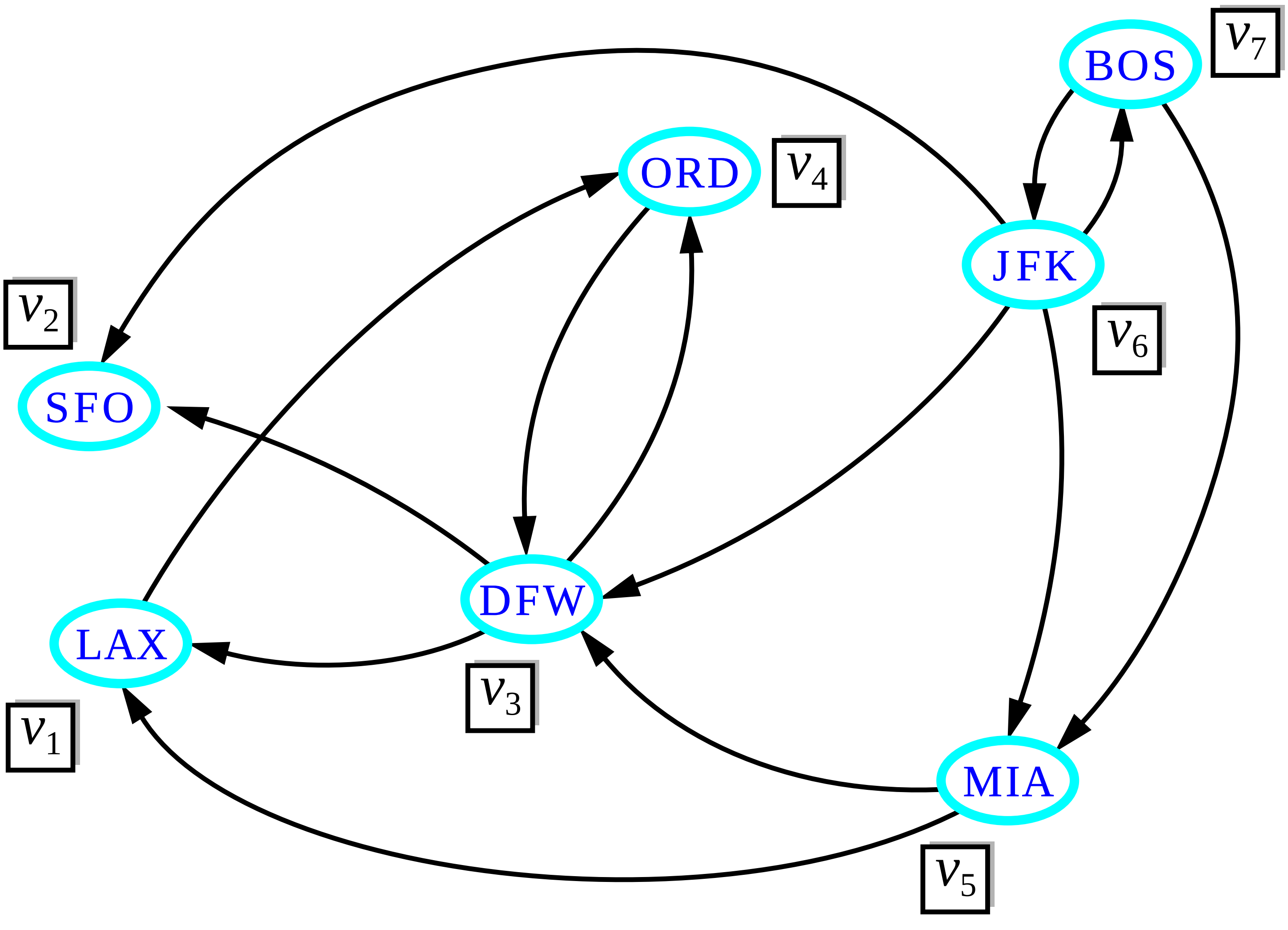
- this algorithms assumes that methods areAdjacent and insertDirectedEdge take O(1) time (e.g., adjacency matrix structure)

```
Algorithm FloydWarshall(G)
  let v_1 ... v_n be an arbitrary ordering of the vertices
  G_0 = G
  for k = 1 to n do
    // consider all possible routing vertices v_k
    G_k = G_k-1 // these are the only ones you need to
                   store
    for each (i, j = 1, ..., n) (i != j) (i, j != k) do
      // for each pair of vertices v_i and v_j
      if G_k-1.areAdjacent(v_i,v_k) and
         G_k-1.areAdjacent(v_k,v_j) then
           G_k.insertDirectedEdge(v_i,v_j,null)
    return G_n
```

- digraph G_k is the subdigraph of the transitive closure of G induced by paths with intermediate vertices in the set $\{ v_1, ..., v_k \}$
- running time: $O(n^3)$

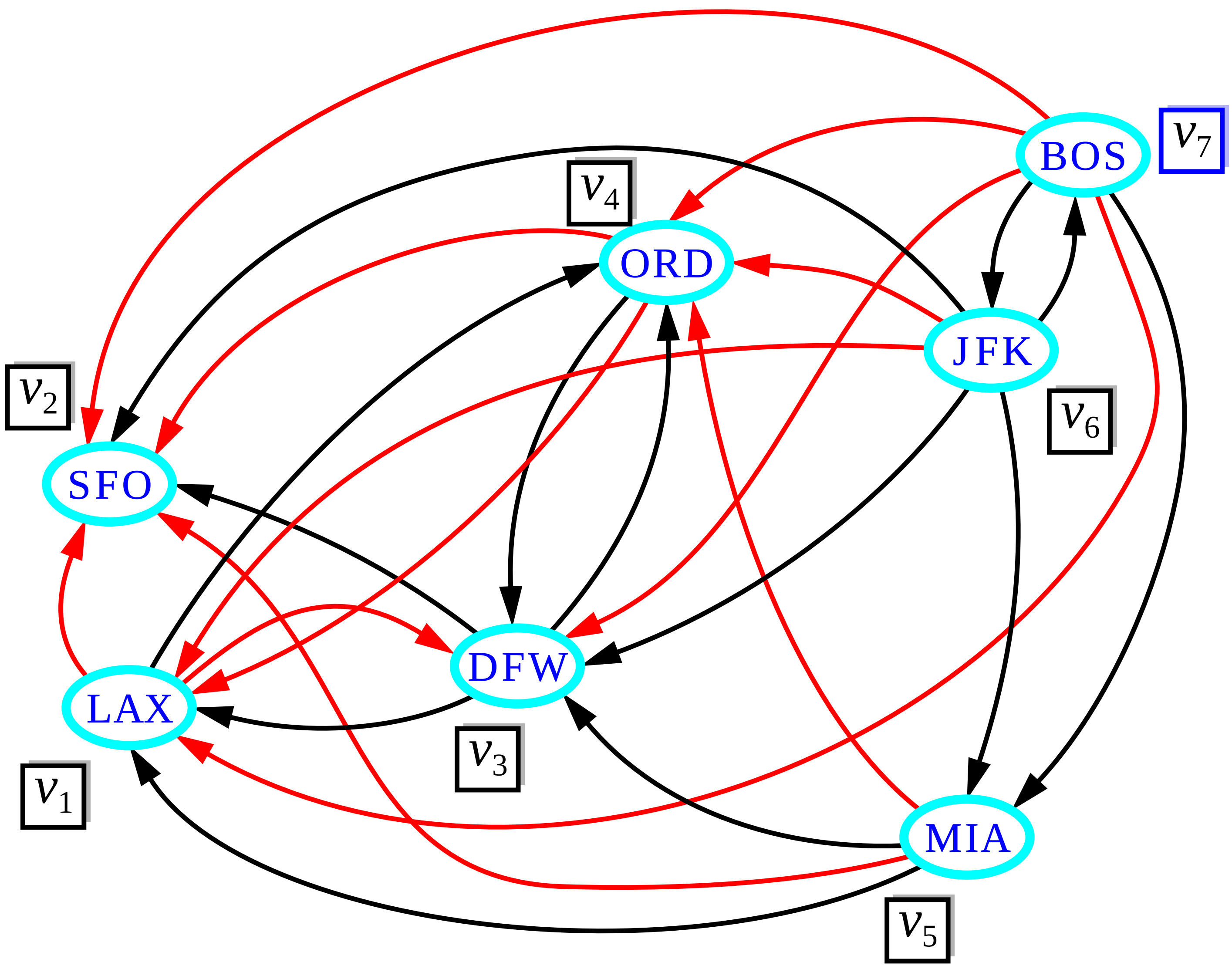
Example

- digraph G



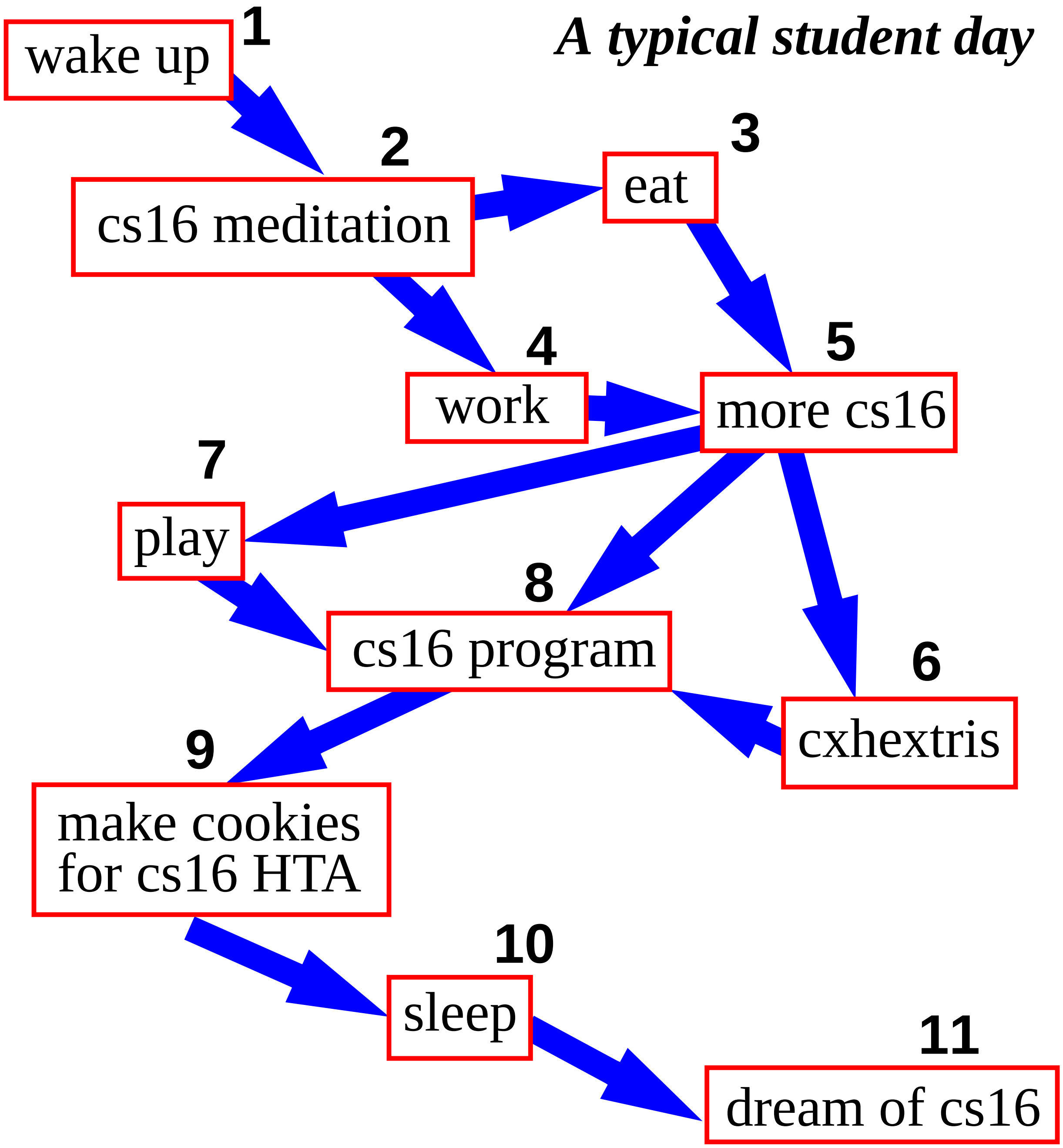
Example

- digraph G*



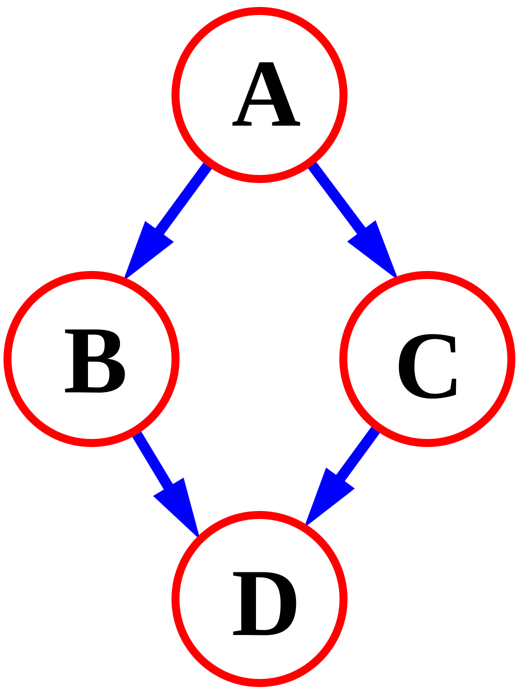
Topological Sorting

For each edge (u,v), vertex u is visited before vertex v



Topological Sorting

Topological sorting may not be unique



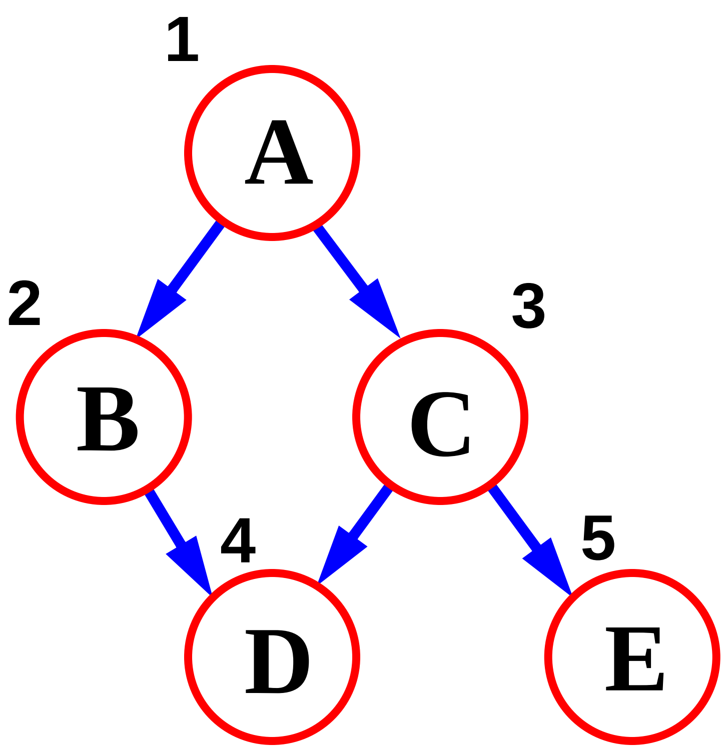
A B C D
or
A C B D

– ***You** make the call!*

Topological Sorting

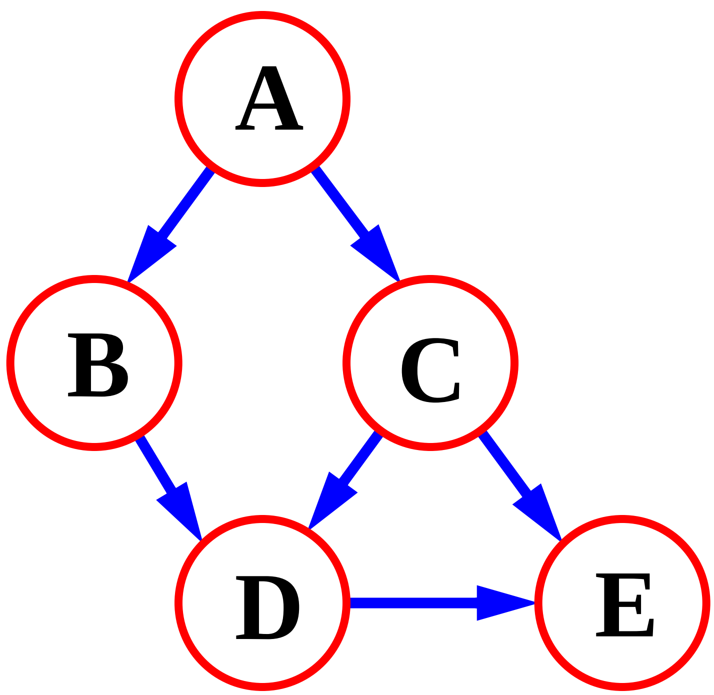
Labels are increasing along a directed path

A digraph has a topological sorting *if and only if* it is acyclic (i.e., a dag)



Algorithm for Topological Sorting

```
method TopologicalSort
   if there are more vertices
         let v be a source;
               // a vertex w/o incoming edges
         label and remove v;
          TopologicalSort;
```



Algorithm (continued)

Simulate deletion of sources using indegree counters

```
TopSort(Vertex v);
   label v;
   foreach edge(v,w)
            indeg(w) = indeg(w) – 1;
            if indeg(w) = 0
                 TopSort(w);
```

1. Compute indeg(**v**) for all vertices
2. Foreach vertex **v** do
   if **v** not labeled and indeg(**v**) = 0
   then TopSort(**v**)

Example