Special Graphs

- Some graph types are used so often that they have special names
  - undirected graph
  - directed graph
  - tree
  - dag (directed acyclic graph)
  
  - acyclic ≡ no cycles
  - cycle ≡ path that starts and ends at the same vertex (i.e., a loop)

Example dag:

Suppose you have a directed graph; how do you tell if it's a dag?

Topological Sort

- Claim: A directed graph is a dag iff it can be topologically sorted

A topological sort of a directed graph is an ordering of the vertices such that if there is an edge from u to v then u appears before v in the ordering.

There are usually multiple solutions for topological sort.

Example solutions:

Algorithm for Topological Sort

count[v] = number of edges into vertex v
W = set of all vertices
while W nonempty:
  Choose u in W with count[u] = 0
  Remove u from W
  for v such that (u, v) is an edge:
    count[v] = count[v] – 1

If the graph is not acyclic then the "Choose" step will fail

Correctness

- We can find a vertex that is acceptable as the first vertex
- Once we remove that vertex (and its edges) we have a new topological sort problem of smaller size

Runtime for Topological Sort

count[v] = number of edges into vertex v
W = set of all vertices
while W nonempty:
  Choose u in W with count[u] = 0
  Remove u from W
  for v such that (u, v) is an edge:
    count[v] = count[v] – 1

If the graph is not acyclic then the "Choose" step will fail

- Adjacency Matrix
  - while loop is executed |V| times
  - Choose step: O(V)
  - Total time: O(V^2)

- Adjacency List
  - while loop is executed |V| times
  - Choose step: O(1)
  - Use a stack of waiting 0-count vertices
  - Decrementing count: O(E) over all
  - Total time: O(E) + |V|