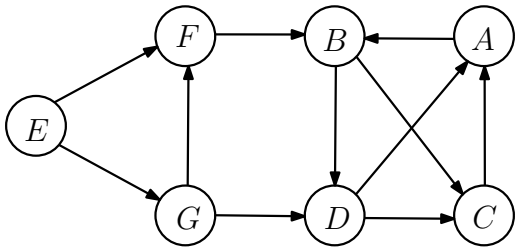
COMP 3804 — Tutorial February 25, 2026

Algorithm DFS(G):
for each vertex v
do $visited(v) = false$
endfor;
$clock = 1$;
for each vertex v
do if $visited(v) = false$
    then EXPLORE(v)
    endif
endfor

Algorithm EXPLORE(v):
$visited(v) = true$;
$pre(v) = clock$;
$clock = clock + 1$;
for each edge (v, u)
do if $visited(u) = false$
    then EXPLORE(u)
    endif
endfor;
$post(v) = clock$;
$clock = clock + 1$

Problem 1: Consider the following directed graph:



(1.1) Draw the *DFS*-forest obtained by running algorithm DFS. Classify each edge as a tree edge, forward edge, back edge, or cross edge. In the *DFS*-forest, give the *pre*- and *post*-number of each vertex. Whenever there is a choice of vertices, pick the one that is alphabetically first.

(1.2) Draw the *DFS*-forest obtained by running algorithm DFS. Classify each edge as a tree edge, forward edge, back edge, or cross edge. In the *DFS*-forest, give the *pre*- and *post*-number of each vertex. Whenever there is a choice of vertices, pick the one that is alphabetically last.

Problem 2: Let $G = (V, E)$ be a directed acyclic graph, and let s and t be two vertices of V.

Describe an algorithm that computes, in $O(|V| + |E|)$ time, the number of directed paths from s to t in G. As always, justify your answer and the running time of your algorithm.

Problem 3: Let $G = (V, E)$ be a directed graph, which is given to you in the adjacency list format. Thus, each vertex u has a list that stores all vertices of the set

$$\{v : (u, v) \in E\}.$$

The backwards graph G_b is obtained from G by replacing each edge (u, v) in G by the edge (v, u). In words, in G_b, we follow the edges of G backwards.

Describe an algorithm that computes, in $O(|V| + |E|)$ time, an adjacency list representation of G_b. As always, justify your answer and the running time of your algorithm.