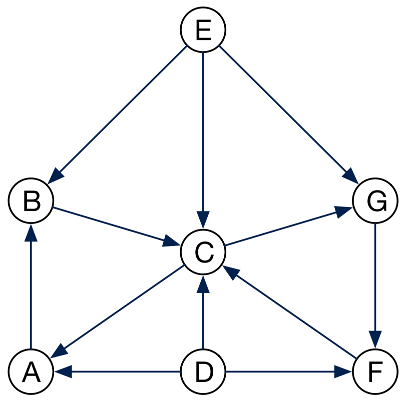
Assignment #8

CMPUT 204
Department of Computing Science
University of Alberta

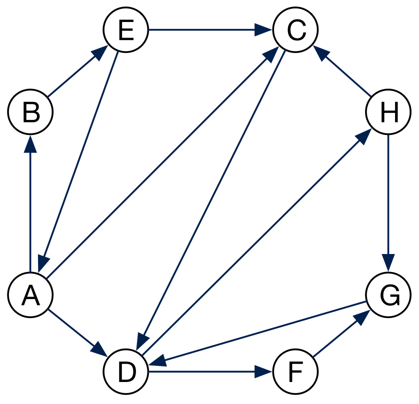
Due: April 4, 2022 (11:59 pm)
(Total 40 marks)

Problem 1. (10 marks)

For the directed graphs given below, perform breath-first search (BFS) and depth-first search (DFS) on the graph using vertex A as a start vertex; whenever there is a choice of vertices, pick the one that is alphabetically first. That is, assume that each adjacency list is ordered alphabetically. Draw the resulting BFS/DFS tree, and classify each edge as a tree edge, forward edge, back edge, or cross edge. For DFS tree, you also need to show the discovery and finish time (i.e., $[dtime, ftime]$) of each node.



(a)



(b)

Problem 2. (10 marks)

Show how to find the strongly connected components (SCCs) of each graph given in Problem 1. Specifically, based on your DFS trees, do the following:

- Show the forest of $G^\intercal$ by traversing nodes in a decreasing $ftime$ in the main DFS loop. (Note that $G^\intercal$ denotes the graph after flipping the direction of each edge in the original graph G)
- Draw the component graph to illustrate the SCCs of the graph, and then given a topological sorting of the component graph.

Problem 3. (20 marks)

We define a *cut vertex* as a node that when removed causes a connected graph to become disconnected, namely, the number of connected components in the graph increases once a cut vertex is removed. For this problem, we will try to find the cut vertices in an undirected graph G with n nodes and m edges.

a. How can we efficiently check whether or not a graph is disconnected?

b. Describe an algorithm that uses a brute force approach to find all the cut vertices in G in $O(n(n+m))$ time. You do not need to write the pseudo code.

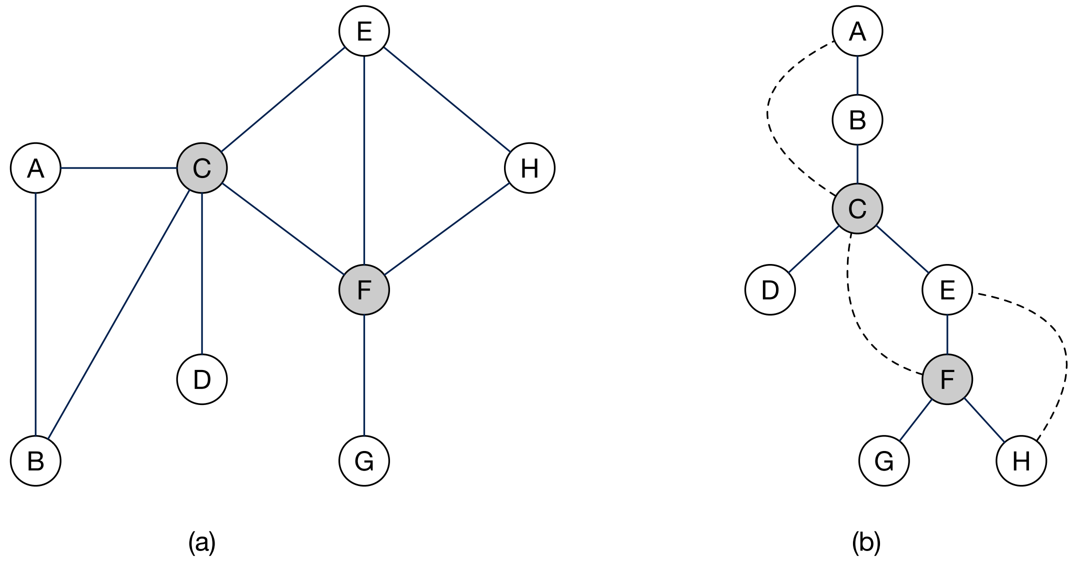


Figure 1: A cut vertex is any vertex whose removal will increases the number of connected components. In subfigure (a), the two gray nodes (i.e., C and F) are cut vertices. Subfigure (b) illustrates the DFS tree of the left graph starting from vertex A, where non-tree edges are shown as dotted lines.

c. Draw two DFS trees starting from vertex C and F (again, whenever there is a choice of vertices, pick the one that is alphabetically first). Indicate non-tree edges using dotted links.

d. Based on the given DFS tree (starting from vertex A) in Figure 1(b) and the two DFS trees in part c (starting from vertex C, F), prove that i) the *root* of a DFS tree is a cut vertex if and only if it has at least two children, and ii) the *leaf* of a DFS tree is never a cut vertex.

e. In a DFS tree, other than the root and leaf nodes, we also have non-root, non-leaf vertices. We argue that any non-root, non-leaf vertex u is a cut vertex *if and only if* there exists a subtree rooted at a child of u that has no back edges to any *ancestor* of u. Note that the claim is an *if and only if* statement, so we need to prove it in both directions. Please follow the partial proof given below to complete the proof of both directions.

Claim 1: If u is a cut vertex, then there exists a subtree rooted at a child of u that has no back edges to any ancestor of u.

Proof of Claim 1: Assume for the sake of contradiction that all the subtrees have back edges to some ancestor of u. If we remove u ...

Claim 2: If there exists a subtree rooted at a child of u that has no back edges to any ancestor of u, then u is a cut vertex.

Proof of Claim 2: Note that a DFS on an undirected graph only produces tree edges and forward/back edges, no cross edges...

Comment: The idea in part d and e can be extended to yield an $O(n + m)$ algorithm to find all cut vertices, improving upon the $O(n(n + m))$ brute force algorithm you proposed in part b. But we will not require you to design the algorithm in this assignment.