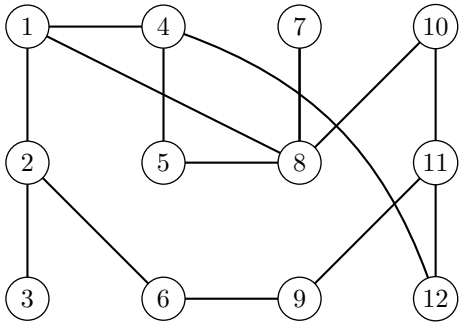
ASSIGNMENT 2 CMSC 451 (Spring 2016)

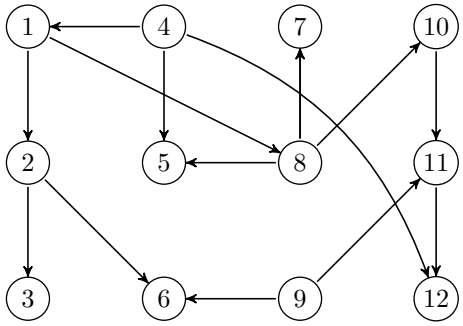
Due at the start of class on Thursday, March 3.

1. *BFS and DFS on undirected and directed graphs.*

   Consider the following two graphs:



G_1



G_2

   (a) Run the breadth-first search algorithm on the undirected graph G_1, starting from vertex 1, and show its final output. When you have to choose which vertex to process next (and that choice is not otherwise specified by the BFS algorithm), use the one with the smallest label. Draw the tree edges with solid lines and the non-tree edges with dashed lines. Indicate the layers of the BFS tree.

   (b) Is G_1 bipartite? Justify your answer with reference to the BFS tree.

   (c) Repeat part (a) on the directed graph G_2.

   (d) Run the depth-first search algorithm on the undirected graph G_1, starting from vertex 1, and show its final output. When you have to choose which vertex to process next (and that choice is not otherwise specified by the DFS algorithm), use the one with the smallest label.

   (e) Repeat part (d) on the directed graph G_2.

   (f) Is G_2 strongly connected? Justify your answer with reference to a search tree (or trees).

2. *When are BFS and DFS the same?*

   Characterize the set of undirected, connected graphs G with the following property: there exists a vertex s of G such that some BFS tree rooted at s is identical to some DFS tree rooted at s. Prove that your characterization is correct.

3. *DAG algorithms.*

   (a) Design an algorithm that, given a directed graph as input, determines whether it is a directed acyclic graph (DAG).

   (b) If we view a DAG as representing precedence constraints on a set of operations at the vertices (i.e., an operation cannot be performed until the operations at the heads of all incoming edges have been performed), the length of a longest path in the DAG represents the amount of time required to perform all the operations, assuming operations can be performed in parallel (subject to the precedence constraints). Design an algorithm that, given a DAG as input, determines the length of a longest path. (Hint: one possible approach is based on depth-first search; another is based on topological sorting. Any correct algorithm is acceptable.)

(c) Again viewing a DAG as encoding precedence constraints on operations that may be performed in parallel, the *earliest start time* of a vertex is the earliest time at which that operation can be performed subject to the precedence constraints (where time starts at 0 and each step of the execution takes unit time). Design an algorithm that, given a DAG as input, computes the earliest start time of every vertex.

In each part, prove the correctness of your algorithm and analyze its running time. For full credit, all your algorithms should run in time $O(n + m)$, where the input graph has n vertices and m edges.

4. *Route planning with variable travel times.*

Suppose you would like to navigate a system of roads, represented by a directed graph $G = (V, E)$. Because of variable traffic and weather conditions, the time required to travel along an edge varies with time. Fortunately, you are given access to an oracle that can reliably predict how long it will take to travel along an edge at any chosen starting time. Given an edge $e = (u, v) \in E$ and a starting time t, the oracle returns the amount of time $r_e(t) \geq 0$ that it will take to travel from vertex u to vertex v along edge e, assuming you leave u at time t. The travel times have the property that if $t' > t$, then $t' + r_e(t') > t + r_e(t)$ (i.e., if you leave later, you will arrive later), but are otherwise arbitrary. Each query to the oracle can be performed in time $O(1)$. Design a polynomial-time algorithm to find the fastest way to get from a given starting vertex to a given destination vertex, starting at time 0. Prove that your algorithm is correct and analyze its running time.