- Please note that one of the main goals of this course is to design efficient algorithms. So, there are points for efficiency even though we may not explicitly state this in the question.
- Unless otherwise mentioned, assume that graphs are given in adjacency list representation.
- In the lectures, we have discussed an $O(V + E)$ algorithm for finding all the SCCs of a directed graph. We can extend this algorithm to design an algorithm `CreateMetaGraph(`G`)` that outputs the meta-graph of a given directed graph in $O(V + E)$ time. For this homework, you may use `CreateMetaGraph(`G`)` as a sub-routine.
- The other instructions are the same as in Homework-0.

There are 5 questions for a total of 100 points.

1. A tree is defined to be an undirected graph that does not contain any cycles. An undirected graph is said to be connected iff for every pair of vertices, there is a path between them. For this question, you have to show the following statement:

   Any connected undirected graph with n vertices and $(n - 1)$ edges is a tree.

   We will prove the statement using Mathematical Induction. The first step in such a proof is to define the propositional function. Fortunately for this problem, this is already given in the statement of the claim.

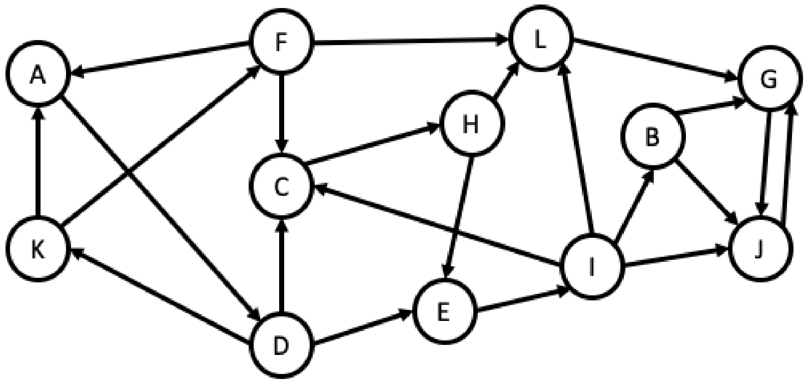
   $P(n)$: Any connected undirected graph with n vertices and $(n - 1)$ edges is a tree.

   The base case is simple. $P(1)$ holds since any graph with 1 node and 0 edges is indeed a tree. For the inductive step, we assume that $P(1), P(2), ..., P(k)$ holds for an arbitrary $k \geq 1$, and then we will show that $P(k + 1)$ holds. Consider any connected graph G with $(k + 1)$ nodes and k edges. You are asked to complete the argument by doing the following:

   (a) (10 points) Show that G has a node v with degree 1.

   (b) (5 points) Consider the graph G' obtained from G by removing vertex v and its edge. Now use the induction assumption on G' to conclude that G is a tree.

2. Consider the following directed graph and answer the questions that follow:



   (a) (1 point) Is the graph a DAG?

   (b) (2 points) How many SCCs does this graph have?

(c) (1 point) How many source SCCs does this graph have?

(d) (2 points) Suppose we run the DFS algorithm on the graph exploring nodes in alphabetical order. Given this, what is the pre-number of vertex F?

(e) (2 points) Suppose we run the DFS algorithm on the graph exploring nodes in alphabetical order. Given this, what is the post-number of vertex J?

(f) (2 points) Is it possible to add a single edge to this graph so that the graph becomes a strongly connected graph? If so, which edge would you add?

3. (25 points) Design an algorithm that takes as input a directed acyclic graph G and determines if there is a directed path that visits every vertex exactly once. Give running time analysis and proof of correctness.

4. (25 points) Given a directed acyclic graph $G = (V, E)$ and a vertex u, design an algorithm that outputs all vertices $S \subseteq V$ such that for all $v \in S$, there is an even length simple path from u to v in G.
(A simple path is a path with all distinct vertices.)

   Give running time analysis and proof of correctness.

5. Given a directed graph $G = (V, E)$ that is not a strongly connected graph, you have to determine if there exists a vertex that is reachable from every vertex in the graph. Design an algorithm for this problem. Your algorithm should output "yes" if such a vertex exists and "no" otherwise.

   (a) (15 points) Give a linear time algorithm that works for DAG's.

   (b) (10 points) Extend this to a linear time algorithm that works for any directed graph. (*Hint: Consider making use of the meta-graph of the given graph.*)

   Give running time analysis and proof of correctness for both parts.