CS 473: Algorithms, Fall 2010
HW 1 (due Tuesday, September 7th)

This homework contains four problems. **Read the instructions for submitting homework on the course webpage**. In particular, *make sure* that you write the solutions for the problems on separate sheets of paper. Write your name and netid on each sheet.

Collaboration Policy: For this home work, Problems 2-4 can be worked in groups of up to 3 students each.

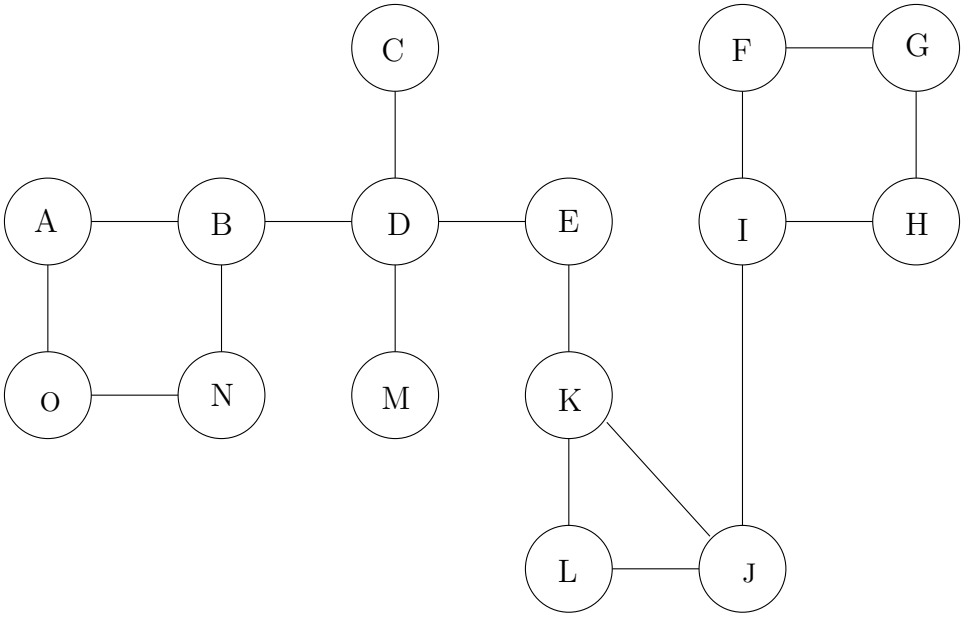
Problem 1 should be answered in Compass as part of the assessment HW1-Online and should be done individually.

1. (15 pts) Short questions to be answered on compass individually.
2. (10 pts) Let s, t be two nodes in an undirected graph G such that the distance between them (in terms of number of edges) is strictly greater than $n/2$; here n is the number of nodes in G. Show that there is a node v such that all s-t paths contain v.
3. (45 pts) Given a connected *undirected* graph $G = (V, E)$, an edge $e = (u, v)$ is called a *bridge*, or a *cut-edge*, if removing e disconnects the graph into two pieces, one containing u and the other containing v. A vertex u is called a *separating vertex*, or *cut-vertex*, if removing u leaves the graph into two or more disconnected pieces; note that u does not count as one of the pieces in this definition. Your goal in this problem is to develop a linear time algorithm to find *all* the bridges and cut-vertices of a given graph using DFS. Let T be a DFS tree of G (note that it is rooted at the first node from which DFS is called). For a node v we will use the notation T_v to denote the sub-tree of T hanging at v (includes v).

   - In the graph shown in the figure, identify all the bridges and cut-vertices.
   - Prove that any bridge of G has to be a tree edge in every DFS(G). Recall the property of DFS in undirected graphs. Why does this show that the maximum number of bridges is $n - 1$? What is a graph that achieves this bound?
   - Suppose $e = (u, v)$ is a tree-edge in DFS(G) with $pre(u) < pre(v)$ (that means u is the ancestor of v). Prove that e is a bridge if and only if (need to prove both directions) there is no edge from any node in T_v to either u or any of its ancestors.
   - For each node u define:

     $$low(u) = \min \begin{cases} pre(u) \\ pre(w) \text{ where } (v, w) \text{ is a back edge for some descendant } v \text{ of } u \end{cases}$$

     Give a linear time algorithm that computes the low value for all nodes by adapting DFS(G). Give the altered pseudo-code of DFS(G) to do this. There is no need to prove that your code is correct.
   - Give a linear time algorithm that identifies *all* the bridges of G using the low values and the steps above. Specifically, provide pseudo-code for a linear time algorithm to do so. There is no need to prove that your code is correct.



- Prove that the root of the DFS tree is a cut-vertex if and only if it has two or more children.
- Prove that a non-root vertex u of the DFS tree T is a cut-vertex if and only if it has a child v such that no node in T_v has a backedge to a *proper* ancestor of u (that is, an ancestor of u which is not u itself).
- The above two properties can be used to find all the cut-vertices in linear time. Give the pseudo-code for a linear time algorithm to do so. There is no need to prove that your code is correct.

It is instructive to run DFS(G) on the example graph and compute the pre values and the low values for each node.

4. (30 pts) Let $G = (V, E)$ be a directed graph. Define a relation R on the nodes V as follows: uRv iff u can reach v or v can reach u.

   (a) (5 pts) Is R an equivalence relation? If yes, give a proof, otherwise give an example to show it is false.

   (b) (25 pts) Call G semi-strongly-connected if for every pair of nodes (u, v), uRv. Give a linear time algorithm to determine if G is semi-strongly-connected. By linear time we mean an algorithm that runs in time $O(m + n)$ where $m = |E|$ and $n = |V|$.