

CS 473: Algorithms, Fall 2008

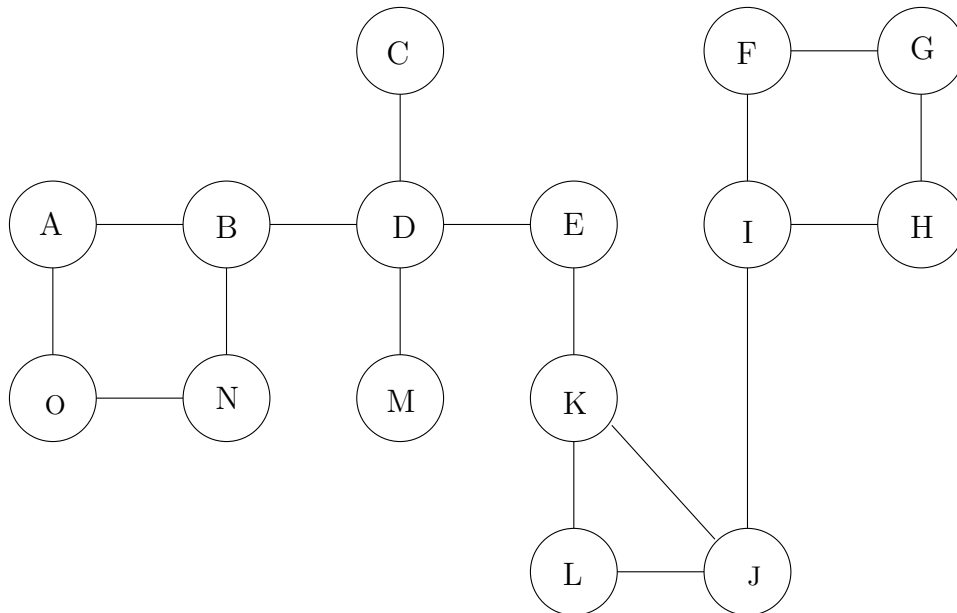
HW 2 (due Tuesday, September 16, 11am)

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

Collaboration Policy: For this homework you are allowed to work in groups of up to 3 students each. Starting this week, a third of the on-campus students will be presenting their homework orally. Please see the newgroup for instructions on which groups will be presenting orally and the instructions for signing up for a slot. The other groups will submit a written homework.

- (20 pts) Problem 3.6 from the text book.
- (50 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 $\text{DFS}(G)$ with $\text{pre}(u) < \text{pre}(v)$ (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:

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

Show that the low value for all nodes can be computed in linear time during $\text{DFS}(G)$. Give the altered pseudo-code of $\text{DFS}(G)$ to do this. There is no need to prove that your code is correct.

- Show how you can identify *all* the bridges of G using the low values and the steps you have shown above. Conclude with a linear time algorithm for the problem.
- Show that the root of the DFS tree is a cut-vertex if and only if it has two or more children.
- Show 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).
- Show how the low values can be used to output all the cut-vertices in linear time.

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

3. (30 pts) A directed graph $G = (V, E)$ is *weakly connected* if for every pair of vertices (u, v) , u can reach v or v can reach u . Note that a strongly connected graph is also weakly connected but not vice-versa. Give a linear time algorithm to test whether a given directed graph is weakly connected or not. Linear time means a running time of $O(m + n)$ where m is the number of edges and n is the number of nodes.