Homework 2 – Due Thursday, September 8, 2016 on Canvas

Please refer to HW guidelines from HW1, course syllabus, and collaboration policy.

Exercises (Do not hand in.) In addition to the exercises in Chapter 3 and in the lecture notes:

- We say a directed graph is simply connected if, for each pair of vertices u and v, either there is a path from u to v or a path from v to u (or both). Give an $O(m + n)$ time algorithm to determine if a given directed graph is simply connected.

Problems to be handed in, 10 points each (Don't forget to prove correctness and analyze time/space requirements of your algorithm.)

1. (**Vulnerable edges**, 2-page limit)
   Let $G = (V, E)$ be an undirected graph with $|V| = n$ and $|E| = m$. An edge $e \in E$ is *vulnerable* if removing e from the graph increases the number of connected components (that is, removing e breaks up one of the graph's connected components).

   (-) Argue (but do not hand it in) that DFS in an undirected graph produces only *tree* and *back* edges (no *forward* or *cross* edges).

   (-) Argue (but do not hand it in) that an edge is vulnerable iff it is not on any cycle.

   Recall that $v.d$ records the time when vertex v is discovered by DFS.

   (a) Consider an edge (u, v), where $u.d < v.d$. Prove that (u, v) is vulnerable iff the following conditions hold (1) (u, v) is a tree edge and (2) there no back edge from v or its descendant in the DFS tree to u or its ancestor in the DFS tree.

   (b) Modify DFS to obtain an $O(n+m)$-time algorithm for finding all the vulnerable edges of a graph given in adjacency list format. *Hint:* For each vertex v, compute the smallest time stamp $w.d$, where w is reachable by a back edge from v or one of its descendants.

2. (**Births and Deaths**, 2-page limit) Chapter 3, problem 12.