CPSC 320 Assignment #3

January 22, 2015

Due date: Monday, 2015/02/02 at 5PM

Staple your solution behind the CPSC 320 cover page and submit in our handin box.

We guarantee that we'll mark at least one part of problem 1, at least one sub-part of each part of problem 2, and (all of) problem 3.

1. DFS in a directed graph works much like DFS in an undirected graph.
   (a) Prove or disprove: In a directed graph with $n \geq 2$ nodes, if DFS run from two different nodes reaches n nodes both times, then the graph has a cycle.
   (b) Prove or disprove: In a directed graph with $n \geq 2$ nodes, if DFS run from two different nodes reaches n nodes both times, then the graph is strongly connected (i.e., is composed of a single strong component).
   (c) We run DFS on a directed graph and draw the tree that results (root at the top, subtrees arranged in the order they are explored from left-to-right). Clearly, edges can go one level down (from a parent to a child). For each of the following other possibilities, prove or disprove that (dashed) edges can go that direction:
      i. One level up.
      ii. More than one level up.
      iii. To the left (i.e., to a subtree left of this node's subtree in some ancestor).

2. In the "minimum edge cover" problem, the input is a connected, undirected graph $G = (V, E)$ with $|V| \geq 2$, and the output is the smallest possible set E' such that $E' \subseteq E$ and for all vertices $v \in V$, there is an edge (v, u) (or equivalently (u, v)) in E'. That is, every vertex in the graph is the endpoint of some edge in E'.

(a) Propose a simple, reasonable greedy algorithm for this problem. The algorithm should find an edge cover but need not necessarily find the **minimum** edge cover.

   i. Write out your algorithm in clear pseudocode.
   ii. Give a good Θ-bound on the worst-case runtime of your algorithm. (Specify any necessary properties of data structures you used!)
   iii. Prove that your algorithm finds an edge cover.
   iv. Either prove that your algorithm is correct (unlikely!) or give an example that shows that your algorithm may not find the smallest edge cover.

(b) A maximum matching in a graph $G = (V, E)$ is the largest set E'' such that $E'' \subseteq E$ and there are no three vertices $v_1, v_2, v_3 \in V$ such that (v_1, v_2) and (v_1, v_3) are in E'' (bearing in mind that the vertex order doesn't matter since in an undirected graph $(x, y) = (y, x)$). That is: E'' "marries off" as many vertices as possible without having any one vertex "married" to two or more vertices.

   In this part, assume you have a graph $G = (V, E)$ and a maximum matching for the graph E''.

   i. Given a maximum matching, give a greedy algorithm to find a minimum edge cover in the graph.
   ii. Give a good Θ-bound on the **best**-case runtime of your algorithm. (Specify any necessary properties of data structures you used!)
   iii. Prove that your algorithm finds an edge cover.
   iv. Prove that your algorithm finds an edge cover that is better than any edge cover that does **not** contain a maximum matching.
   v. Prove that your algorithm finds a minimum edge cover.

3. Problem 4.3 in the text. (That's the one about a trucking company shipping packages between New York and Boston.) However, we recommend trying this both as a "stays ahead" and an "exchange" argument. (Clearly indicate your "stays ahead" argument, since that's the only one we'll mark.)