CSCI 3104-Spring 2016: Assignment #6.

Assigned date: Wednesday, 3/30/2016,
Due date: Tuesday, 4/5/2016, before class
Maximum Points: 40 points (includes 5 points for legibility).

Note: This assignment *must be turned in on paper, before class.* Please do not email: it is very hard for us to keep track of email submissions. Further instructions are on the moodle page.

P1 (20 points) Let G be an undirected graph with n nodes and m edges stored as an adjacency list. Recall that G is bipartite (or two colorable) if and only if we can color each vertex either red or blue so that no edges go between vertices of the same color.

Suppose we have run a DFS on G, yielding a DFS tree T. Describe how we can check if G is bipartite? (Hint: use the DFS tree to assign colors and check that the forward, back and cross edges satisfy the bipartite condition above).

Solution.

1. Step one is to assign colors to the nodes in the graph according to the tree.
2. Let root color be red.
3. Traverse the tree starting from root.
4. For each node if the parent is colored red, color the node blue and vice-versa.
5. Next, go through all edges in the graph and check if the edge connects two nodes of different colors.
6. If all edges go to nodes of different colors then we have a bipartite graph. Otherwise we do not.

P2 (5 points) Find a solution for the following system of inequalities over unknown numbers x_1, x_2, x_3, x_4, x_5.

$$x_1 < x_3,\ x_1 < x_4,\ x_2 < x_5,\ x_5 < x_4$$

Solution. $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 5, x_5 = 4$.

P3 (15 points) Complex tasks like building a large bridge (or a software system) are often broken down into a series of simpler subtasks. Let $J_1, \ldots, J_n$ be n different subtasks. There are often constraints that say that a particular task cannot be started before another is completed.

Suppose you are given a set of tasks and a list of requirements where each requirement R_j says that Job J_i cannot be started before job J_k is finished. We represent each requirement as a pair (i, k).

Write an algorithm that given a set of jobs and requirements will find out if the jobs can be scheduled in order to meet the requirements. Also, if the jobs are possible to schedule, then your algorithm should print one possible ordering of completing the jobs while meeting all the requirements.

Solution. The solution is to perform a topological sort of the graph whose vertices are the jobs and edges denote the requirements. First perform a DFS traversal of the graph. If there are any back edges, we conclude a cycle and the jobs cannot be completed. Otherwise, we sort the nodes in the graph with decreasing order of their finish times to yield a task ordering.