FALL 2008: **COT 5407** INTRO. TO ALGORITHMS

[HOMEWORK 6; DUE DEC 2 (NOT NOV 30) AT START OF CLASS]

General submission guidelines and policies: ADD THE FOLLOWING SIGNED STATEMENT. Without this statement, your homework will not be graded.

> I HAVE ADHERED TO THE COLLABORATION POLICY FOR THIS CLASS. IN OTHER WORDS, EVERYTHING WRITTEN DOWN IN THIS SUBMISSION IS MY OWN WORK. FOR PROBLEMS WHERE I RECEIVED ANY HELP, I HAVE CITED THE SOURCE, AND/OR NAMED THE COLLABORATOR.

Read the handout on **Homework guidelines and collaboration policy**.

Problems

31. (**Regular**) Modify DFS or BFS to design an algorithm called CHECKFORODDCYCLE for checking whether a given connected, undirected, unweighted, simple graph $G(V, E)$ has an odd length cycle. Assume that the input graph has n vertices and m edges. Provide an argument why you think the algorithm is correct and analyze its time complexity.

32. (**Regular**) Design an efficient algorithm to compute the in-degree and out-degree of every vertex in a given directed graph $G(V, E)$. The in-degree (out-degree, resp.) of a vertex is defined as the number of edges directed into (out of, resp.) that vertex. Assume that the input graph has n vertices and m edges.

33. (**Exercise**) Design an efficient algorithm that takes a directed graph $G(V, E)$ as input and outputs a directed graph $G'(V, E')$ where every edge is reversed. In other words, for every edge $e = (u, v) \in E$, there is an edge $e' = (v, u) \in E'$ and vice versa. Assume that the input graph has n vertices and m edges.

34. (**Regular**) You are given a "probabilistic" directed graph $G(V, E)$, where each directed edge e is associated with the probability of taking that edge and is denoted by $p(e)$. Also, assume that the probability of taking a path is simply the product of the probabilities of taking each of the edges on that path. Given a source vertex s, design an algorithm that computes the probabilities of the most probable simple paths to every vertex. Recall that a simple path is one that does not revisit any vertices.

35. (**Exercise**) Given a weighted undirected graph G with non-negative edge weights, if the edge weights are all increased by a positive additive constant, can the minimum spanning tree change? Can the output of Dijkstra's algorithm change for some (fixed) start vertex s? What if they are decreased by a positive constant? What if the edge weights are all multiplied by a positive constant? Give (very) simple examples, if you claim that they can change.

36. (**Exercise**) Does Dijkstra's algorithm work correctly if some edge weights are negative? Does it work correctly if some edge weights are negative, but there are no negative weight cycles?

37. (**Regular**) Can Dijkstra's algorithm work be modified to work correctly if all edges have non-negative weights, but we need to compute the longest simple path from the source to every vertex?

38. (**Extra Credit**) Problem 23.2-7, page 574.

39. (**Extra Credit**) Problem 23-3, page 577.

40. (**Regular**) Modify Floyd-Warshall's algorithm to output the number of distinct shortest paths between every pair of vertices in an unweighted undirected graph.