SPRING 2017: **COT 5407** Intro. to Algorithms

[Homework 7; Due April 22 via Email]

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** from your course website before you start on this homework. This is very important.
You only need to submit solutions to problems marked (**Regular**). All others are optional.

Problems

52. (**Exercise**) 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.

53. (**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.

54. (**Exercise**) Problem 22.2-7, page 602.

54. (**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.

55. (**Regular**) In the MST problem, we are required to find a spanning tree that has the minimum sum of weights of its edges. In the Miami version of the problem, MMST, we are required to minimize the largest edge weight. Design an algorithm to compute an MMST. Aruge that it is correct. Analyze its time complexity.

56. (**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. Analyze the time complexity of your algorithm.

57. (**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.

58. (**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?

59. (**Exercise**) 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?

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

61. (**Exercise**) Problem 23.2-7, page 637.

62. (**Exercise**) Problem 23-3, page 640.