CS420/520: Graph Theory with Applications to CS, Winter 2017

Homework 4

Due: Tue, Feb/14/17

Homework Policy:

1. Students should work on homework assignments in groups of preferably three people. Each group submits to TEACH one set of typeset solutions, and hands in a printed hard copy in class or slides the hard copy under my door before the midnight of the due day. The hard copy will be graded.
2. The goal of the homework assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
3. You are allowed to discuss the problems with other groups, and you are allowed to use other resources, but you *must* cite them. Also, you must write everything in your own words, copying verbatim is plagiarism.
4. *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a completely wrong answer will receive zero.
5. Algorithms should be explained in plain english. Of course, you can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.

Readings:

(A) Jeff lecture notes on all pairs shortest paths: http://jeffe.cs.illinois.edu/teaching/algorithms/notes/22-apsp.pdf.

Problem 1. Let $G = (V, E)$ be a directed graph with weighted edges; edge weights could be positive, negative, or zero.

(a) How could we delete an arbitrary vertex v from this graph, without changing the shortest-path distance between any other pair of vertices? Describe an algorithm that constructs a directed graph $G' = (V', E')$ with weighted edges, where $V' = V \backslash \{v\}$, and the shortest-path distance between any two nodes in G' is equal to the shortest-path distance between the same two nodes in G, in $O(V^2)$ time.

(b) Now suppose we have already computed all shortest-path distances in G'. Describe an algorithm to compute the shortest-path distances from v to every other vertex, and from every other vertex to v, in the original graph G, in $O(V^2)$ time.

(c) Combine parts (a) and (b) into another all-pairs shortest path algorithm that runs in $O(V^3)$ time. (The resulting algorithm is not the same as Floyd-Warshall!)