CS420/520: Graph Theory with Applications to CS, Winter 2016

Homework 2

Due: Thr, Feb/11/16

Homework Policy: Each student should submit his/her own set of solutions, independently. You are allowed to discuss the homework with other students, however, you need to indicate their names in your submission. Please typeset your solutions.

Readings:

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

(B) Jeff lecture notes on minimum spanning trees: http://jeffe.cs.illinois.edu/teaching/algorithms/notes/20-mst.pdf.

(C) Jeff lecture notes on matroids: http://jeffe.cs.illinois.edu/teaching/algorithms/notes/08-matroids.pdf.

(D) Uri Zwick's lecture notes on matching: http://www.cs.tau.ac.il/~zwick/grad-algo-13/match.pdf.

Problems for practice.

- Problems (3), (4), (6) from (A).
- Problems (7), (8), (9) from (B).
- Problems (1), (3) from (C).
- Problems (1), (2), (4) from (D).

Problem 1. Let $G = (V, E)$ be a directed graph with weighted edges; edge weights could be positive, negative, or zero, but there is no negative cycle. Suppose the vertices of G are partitioned into k disjoint subsets $V_1, V_2, \ldots, V_k$; that is, every vertex of G belongs to exactly one subset V_i. For each i and j, let $\delta(i, j)$ denote the minimum shortest-path distance between vertices in V_i and vertices in V_j:

$$\delta(i,j) = \min\{dist(u,v) | u \in V_i, v \in V_j\}.$$

Describe and analyze an algorithm to compute $\delta(i, j)$ for all $i, j \in \{1, 2, \ldots, k\}$ in time $O(V^2 + kE \log E)$.

Problem 2. Describe and analyze and algorithm to find the second smallest spanning tree of a given graph G, that is, the spanning tree of G with smallest total weight except for the minimum spanning tree.

Problem 3. Prove that for any graph G, the 'matching matroid' of G is in fact a matroid. [Hint: What is the symmetric difference of two matchings?]

Problem 4. Let M be a maximal matching and M^* be a maximum matching. Prove that $|M| \geq |M^*|/2$. Conclude an $O(m)$ time 2-approximation algorithm for computing the maximum matching, an $O(m)$ time algorithm that computes a matching of size at least $1/2$ of the maximum matching.