CMPT 706 — Algorithms for Big Data

Exercises on Graph Algorithms. Due: Thursday, February 28th (at the beginning of the class)

Reminder: the work you submit must be your own. Any collaboration and consulting outside resources must be explicitly mentioned on your submission.

1. There is a natural intuition that two nodes that are far apart in a communication network — separated by many hops — have a more tenuous connection than two nodes that are close together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here is one that involves the susceptibility of paths to the deletion of nodes.

   Suppose that an n-node undirected graph $G = (V, E)$ contains two nodes s and t such that the distance between s and t is strictly greater than $n/2$. Show that there must exist some node v, not equal to either s or t, such that deleting v from G destroys all $s - t$ paths. (In other words, the graph obtained from G by deleting v contains no path from s to t.) Give an algorithm with running time $O(m + n)$ to find such a node v.

2. In social networks analysis it is sometimes useful to find not only the distance between two vertices in a graph or a shortest path between them, but the number of such paths. It turns out this problem can be solved efficiently. More precisely, given an undirected graph (no lengths) $G = (V, E)$ with $|V| = n$ and $|E| = m$, and two vertices $v, w \in V$, suggest an algorithm that outputs the number of shortest $v - w$-paths in G. (The algorithm should not list all the paths, just the number will do.) The running time of your algorithm should be $O(m + n)$.

3. Let $G = (V, E)$ be a directed graph. A *topological order* on G is an ordering of its vertices $V = \{v_1, \ldots, v_n\}$ such that every arc $(v_i, v_j) \in E$ satisfies the condition $i < j$, that is, all arcs are directed from nodes earlier in the order to nodes later in the order.

   (a) Prove that any digraph that admits a topological order is acyclic (or DAG, directed acyclic graph), that is, it does not have directed cycles.

   (b) Suggest an algorithm that given a DAG outputs a topological order,

   (c) Suggest an algorithm that given an arbitrary directed graph G outputs a topological order on G if G is a DAG, or outputs a directed cycle in G, if G is not a DAG.

   Your algorithms should have running time $O(m + n)$.

4. In cases where there are several different shortest paths between two nodes (and edges have varying lengths), the most convenient of these paths is often the one with fewest edges. Accordingly, for a specific starting node s, define

   $\mathsf{best}[u]$ = minimum number of edges in a shortest path from s to u.

   Give an efficient algorithm for the following problem

Input: Graph $G = (V, E)$, positive edge lengths ℓ_e, starting node $s \in V$.

Output: The values of $\mathsf{best}[u]$ for all vertices $u \in V$.

5. You are given a directed graph with (possibly negative) weighted edges, in which the shortest path between any two vertices is guaranteed to have at most k edges. Give an algorithm that finds the shortest path between two vertices u and v in $O(k|E|)$ time.