CS267 HOMEWORK 3 **Due: Nov 13**
Submit by email to cs267stanford@gmail.com Handed out: November 3, 2016

Instructions: Everyone needs to submit their own write-up. If you work together with other students, indicate their names on your write-up.

Problem 1

In this problem we will design a decremental algorithm for single source shortest paths (SSSP) in undirected unweighted graphs. For decremental SSSP, we are given an input graph $G = (V, E)$ and a source node s. The algorithm needs to support queries of the form, given v, report the distance $d(s, v)$. The algorithm needs to support only edge deletions (no insertions).

The preprocessing part of the algorithm will compute a BFS tree T_s from s. As usual, for every node $v \in V$, if s and v are connected, v will be in some level L_i of T_s, where $i = d(s, v)$. Recall, $L_0 = \{s\}$.

The BFS procedure will also compute for each node v three sets, as follows. Let $v \in L_i$ and let $N(v)$ be the set of neighbors of v. Then the sets are $N_1(v) = N(v) \cap L_{i-1}$, $N_2(v) = N(v) \cap L_i$ and $N_3(v) = N(v) \cap L_{i+1}$. (By the properties of BFS, $N_1(v), N_2(v), N_3(v)$ are disjoint and $N_1(v) \cup N_2(v) \cup N_3(v) = N(v)$.)

The decremental algorithm will support deletions of edges while updating the $N_i(v)$ sets of some vertices and moving nodes between levels of the BFS tree. The query time of the algorithm will be constant since for each v, one only needs to query the level of v.

Now consider what happens when an edge (u, v) is deleted from the graph. If $u, v \in L_i$, then removing (u, v) does not change any distances in the graph, so the algorithm only needs to remove u from $N_2(v)$ and v from $N_2(u)$. Suppose that $u \in L_{i-1}$ and $v \in L_i$ (the other case is symmetric). Here we need to remove u from $N_1(v)$ and v from $N_3(u)$. If $N_1(v)$ is nonempty, then again no distances are changed. However, if now $N_1(v) = \emptyset$, v may need to drop to a new level. Moreover, if v drops to a new level, all nodes x that had v in their $N_1(x)$ may need to drop and so on.

(a) Design a recursive procedure drop(x) that given a node x with $N_1(x) = \emptyset$, moves x to the correct level of T_s, updates the sets of the neighbors of x and recursively drops nodes whose distances changed due to the drop of x. Write pseudocode for drop(x) and delete(u, v).

(b) Argue that if the input graph has m edges and n nodes, up until all m edges are deleted, the algorithm performs at most $O(mn)$ operations.

(c) Suppose that instead of keeping a full BFS tree from s, one only keeps a BFS tree with d levels, i.e. only the distances of nodes v with $d(s, v) \leq d$ are maintained. Show that then the total update time of the algorithm would be $O(md)$.

Problem 2

Consider the decremental APSP problem: given an unweighted undirected graph $G = (V, E)$, preprocess it so that one can perform edge deletions, and queries of the form, given u, v, what is the distance between u and v in the current graph.

Show how to use the algorithm from problem 1 to obtain a randomized decremental APSP algorithm with total update time $\tilde{O}(mn^{1.5})$ and query time $\tilde{O}(\sqrt{n})$. The queries should be correct with high probability, provided that the updates are oblivious, i.e. they do not know the random bits used by the algorithm. Give pseudocode for the algorithm, prove its correctness and its running time.

Hint: Use homework 2, problem 3.