CS 473: Fundamental Algorithms, Fall 2014
HW 2 (due Tuesday, at noon, September 16, 2014)

Version: 1.01.

This homework contains three problems. **Read the instructions for submitting homework on the course webpage**.

Collaboration Policy: For this homework, Problems 1–3 can be worked in groups of up to three students.

> Each student individually have to also do **quiz 2** online.

---

1. (30 PTS.) I Want (One) More!
Suppose you are given a directed graph $G = (V, E)$ with non-negative edge lengths; $\ell(e)$ is the length of $e \in E$. You are interested in the shortest path distance between two given locations/nodes s and t. It has been noticed that the existing shortest path distance between s and t in G is not satisfactory and there is a proposal to add exactly one edge to the graph to improve the situation. The candidate edges from which one has to be chosen is given by $E' = \{e_1, e_2, \ldots, e_k\}$ and you can assume that $E \cap E' = \emptyset$. The length of the e_i is $\alpha_i \geq 0$. Your goal is figure out which of these k edges will result in the most reduction in the shortest path distance from s to t. Describe an algorithm for this problem that runs in time $O(n \log n + m + k)$ where $m = |E|$ and $n = |V|$.
(Note that one can easily solve this problem in $O(k(m + n) \log n)$ by running Dijkstra's algorithm k times, one for each G_i where G_i is the graph obtained by adding e_i to G.)

2. (35 PTS.) Walks with at least k distinct nodes.
Given a directed graph $G = (V, E)$ and two nodes s, t, an s-t *walk* is a sequence of nodes $s = v_0, v_1, \ldots, v_k = t$ where (v_i, v_{i+1}) is an edge of G for $0 \leq i < k$. Note that a node may be visited multiple times in a walk — this is how it differs from a path. Given G, s, t and an integer $k \leq n$, design a linear time algorithm to check if there is an s-t walk in G that visits at least k *distinct* nodes including s and t.

(A) (15 PTS.) Solve the problem when G is a DAG.
(B) (20 PTS.) Solve the problem when G is a an arbitrary directed graph. *Hint:* If G is strongly connected then there is always such a walk even for $k = n$ (do you see why?).

3. (35 PTS.) Decreasing/Increasing Weights
Let $G = (V, E)$ be a directed graph with edge lengths that can be negative. Let $\ell(e)$ denote the length of edge $e \in E$ and assume it is an integer. Assume you have a shortest path tree T rooted at a source node s that contains all the nodes in V. You also have the distance values $d(s, u)$ for each $u \in V$ in an array (thus, you can access the distance from s to u in $O(1)$ time). Note that the existence of T implies that G does not have a negative length cycle.
(A) (18 PTS.) Let $e = (p, q)$ be an edge of G that is *not* in T. Show how to compute in $O(1)$ time the smallest integer amount by which we can decrease $\ell(e)$ before T is not a valid shortest path tree in G.

(B) (17 PTS.) Let $e = (p, q)$ be an edge in the tree T. Show how to compute in $O(m + n)$ time the smallest integer amount by which we can increase $\ell(e)$ such that T is no longer a valid shortest path tree. Your algorithm should output ∞ if no amount of increase will change the shortest path tree.