

# CS 473: Algorithms, Fall 2008

## HW 3 (due Tuesday, September 23, 11am)

This homework contains four problems. **Read the instruction for submitting homework on the course webpage.** In particular, *make sure* that you write the solutions for the problems on separate sheets of paper and then staple them together. Write your name and netid on each sheet.

**Collaboration Policy:** For this homework you are allowed to work in groups of up to 3 students each. Starting this week, a third of the on-campus students will be presenting their homework orally. Please see the newgroup for instructions on which groups will be presenting orally and the instructions for signing up for a slot. The other groups will submit a written homework.

1. (10 pts) Problem 3.8 from the text book.
2. (30 pts) 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  $E' = \{e_1, e_2, \dots, e_k\}$  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  $O((m+n) \log n + k)$  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  $G$ .
3. (25 pts) You are given a directed graph  $G = (V, E)$  with potentially negative edge lengths. Your friend ran Dijkstra's algorithm and came up with a shortest path tree  $T$  for distances from a node  $s$ . You realize that Dijkstra's algorithm may not output distances correctly when a graph has negative edge lengths. However, before you run the more expensive Bellman-Ford algorithm, you wish to check whether  $T$  is a correct shortest path tree or not. Describe an  $O(m+n)$  time algorithm to do this check. Make sure you prove that your algorithm is correct.
4. (35pts) Given a directed graph  $G = (V, E)$  and two nodes  $s, t$ , an  $s$ - $t$  walk is a sequence of nodes  $s = v_0, v_1, \dots, v_k = t$  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  $t$ .
  - Solve the problem when  $G$  is a DAG. *Hint:* Can you find a longest path in a DAG?
  - Solve the problem when  $G$  is 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?).