CSOR W4231.002 – Spring, 2018

Homework 4

Out: Monday, March 19, 2018
Due: 8pm, Monday, April 2, 2018

For all algorithms you design, you should always describe them clearly in English, give pseudocode, prove correctness and give the best upper bound that you can for the running time. If you give a dynamic programming algorithm, you should clearly define the subproblems, give the recurrence, analyze time and space requirements and explain how to fill in the dynamic programming table.

Collaboration is limited to discussion of ideas only. You should write up the solutions entirely on your own. You should list your collaborators on your write-up. If you do not type your solutions, be sure that your hand-writing is legible, your scan is high-quality and your name is clearly written on your homework.

You may not use any external resources such as the internet or other textbooks to solve this homework. You should adhere to the department's honesty policy (see the course website, too).

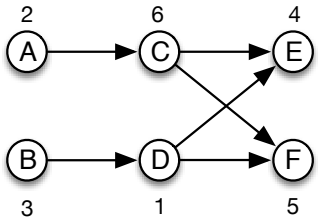
1. (10 points) Give an efficient algorithm to find an odd-length cycle in a directed graph.

2. (10 points) Given an undirected graph $G = (V, E)$ and a specific edge $e \in E$, give an efficient algorithm that determines whether G has a cycle that contains e.

3. (20 points) You are given a directed graph in which each node $u \in V$ has an associated price p_u which is a positive integer. Define the array `cost` as follows: for each $u \in V$,

   $$\texttt{cost}[u] = \text{price of the cheapest node reachable from } u \text{ (including } u \text{ itself)}$$

   For instance, in the graph below (with prices shown for each vertex), the `cost` values of the nodes A, B, C, D, E, F are $2, 1, 4, 1, 4, 5$ respectively.



   Your goal is to design an algorithm that fills in the entire `cost` array.

   (a) Give a linear-time algorithm that works for directed *acyclic* graphs.

   (b) Extend this to a linear-time algorithm that works for all directed graphs.

4. (20 points) In cases where there are several shortest paths between two nodes (and edges have varying lengths), the most convenient among these paths is often the one with the *fewest edges*. We define

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

Give an efficient algorithm that, on input a graph $G = (V, E, w)$ with non-negative edge weights and an origin node $s \in V$, computes $best[u]$ for all $u \in V$.

5. (20 points) Consider a network of roads $G = (V, E)$ connecting a set of cities V. Each road in E has an associated length ℓ_e. There is a proposal to add one new road to this network, and there is a list E' of pairs of cities between which the new road can be built. Each such potential road $e' \in E'$ has an associated length.

As a designer for the public works department you are asked to determine the road $e' \in E'$ whose addition to the existing network G will result in the maximum decrease in the driving distance between two fixed cities s and t in the network. Give an efficient algorithm for solving this problem.

6. (20 points) Problem 24-3, part a. only, from your textbook (p. 679).

Recommended exercises: do NOT return, they will not be graded

1. Problem 22-2, parts a, b, c, d, e and f, in your textbook (p. 622).
2. Problem 22-3 in your textbook (p. 623).