CSOR W4231.001 – Spring, 2020
Eleni Drinea

Homework 4 (80 points)

Out: Thursday, March 26, 2020
Due: 11:59pm, Monday, April 6, 2020

You should always describe your algorithm clearly in English. You should always give pseudocode for all algorithms that you design.

For all deterministic algorithms you design, you must argue correctness and give the best upper bound that you can for the running time. You are encouraged to analyze the space required by your algorithm.

You must submit your assignment as a pdf file. Other file formats will not be graded, and will automatically receive a score of 0. I recommend you type your solutions using LaTeX: you will earn 5 extra credit points if you type your solutions. If you do not type your solutions, make sure that your hand-writing is very neat, your scan is high-quality and your name and UNI are clearly written on your homework.

You should write up the solutions **entirely on your own**. Collaboration is limited to discussion of ideas only. You should adhere to the department's academic honesty policy (see course syllabus).

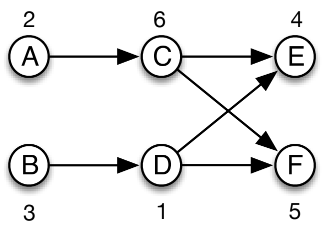
1. (10 points) Given an undirected graph $G = (V, E)$ and a specific edge $e \in E$, give an efficient algorithm that determines whether G has a cycle that contains e.

2. (20 points) Give an efficient algorithm to find an odd-length cycle in a directed graph $G = (V, E)$; or output **no**, if no such cycle exists. *Hint: First solve the problem assuming that the graph is strongly connected.*

3. (30 points) You are given a directed graph $G = (V, E)$ 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 array `cost`.

(a) (15 points) Give a linear-time algorithm that works for directed *acyclic* graphs.

(b) (15 points) 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 $u \in V$.

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).