CS/ECE 374 A (Spring 2022)
Homework 9 (due April 7 Thursday at 10am)

Instructions: As in previous homeworks.

Problem 9.1: We are given a weighted DAG (directed acyclic graph) G with n vertices and m edges with $m \geq n$, where each edge weight may be positive or negative (you may assume that no edge has weight zero). We are also given two vertices $s, t \in V$.

(a) (35 points) Describe an efficient algorithm to determine whether there exists a path from s to t such that the number of positive-weight edges is strictly more than the number of negative-weight edges in the path.

[Hint: there is an $O(m+n)$-time solution (but some partial credit will still be given for an $O(mn)$-time solution). One approach is to use dynamic programming, but a simpler approach is to just run a known algorithm from class on a new weighted graph.]

(b) (65 points) Describe an efficient algorithm for determining whether there exists a path from s to t such that the number of positive-weight edges is strictly more than the number of negative-weight edges in the path *and* the total weight of the path is negative.

[Hint: there is an $O(mn)$-time solution. One approach is to use dynamic programming; another approach is to run a known algorithm on a new graph.]

Problem 9.2: We are given a weighted directed graph $G = (V, E)$ with n vertices, where all edge weights are positive. Each edge is colored red or blue. We are also given an integer $k \leq n$.

We want to compute the shortest closed walk that *contains at least one blue edge and does not have k consecutive red edges.* Describe an efficient algorithm to solve this problem.

(For example, if $k = 4$, a walk with color sequence blue-red-red-blue-red-red-red-blue-blue-red-red-blue is allowed, but not blue-red-red-blue-red-red-red-red-blue. For motivation, imagine that traveling along blue edges lets you recharge. We don't want to travel too long without using a blue edge.)

[Hint: it might be helpful to solve the following all-pairs variant of the problem first: for every pair $u, v \in V$, find the shortest walk from u to v that does not have k consecutive red edges. One approach is to define a new graph and run a known algorithm on the graph.]

[Note: a correct solution with $O(k^2n^3)$ time will get you 90 points; a correct solution with $O(kn^3 \log n)$ or $O(kn^3)$ time will get you 100 points (full credit); and a solution with $O(n^3 \log n)$ time or better will receive 15 more bonus points!]