HW 9

Due on **Wednesday, November 6, 2024 at 9am CST**

CS/ECE 374A: Intro. Algorithms & Models of Computation, Fall 2024 Version: **1.1**

Submission instructions as in previous homeworks.

17 (100 PTS.) Avoiding negativity.

Let G be a directed graph with n vertices and m edges, with weights $w(\cdot)$ on the edges (weights on edges [here] can be any real number, including negative numbers). You are also given a start vertex s. Describe[1] an algorithm that outputs all the vertices x in G, such that *all* the walks from s to x in G do *not* contain a negative cycle.

18 (100 PTS.) Controlled negativity.

Let G be a directed graph with n vertices and m edges, with weights $w(\cdot)$ on the edges.

18.A. (50 PTS.) Assume that for every vertex $v \in \mathsf{V}(\mathsf{G})$ has a start price $\alpha(v)$ (not necessarily positive), and the weights on the edges are all positive. Describe[2] an algorithm that computes the length of the shortest path that ends at x (for all $x \in \mathsf{V}(\mathsf{G})$). Here, a path π that starts at a vertex v and ends at x has ***length***

$$L(v, \pi) = \alpha(v) + w(\pi),$$

where $w(\pi)$ is the total weight of the edges of π. (Hint: First consider the case that $\alpha(\cdot)$ is strictly positive for all the vertices.) Prove that your algorithm is correct.

18.B. (50 PTS.) Now, consider the variant where the weights on the edges can be negative or positive (but there is no start prices on the vertices). You are given in addition to G, two vertices $s, t \in \mathsf{V}(\mathsf{G})$, and a parameter k. Describe[3] an algorithm that computes the shortest walk in G between s and t, where your walk is allowed to travel on at most k negative edges. What is the running time of your algorithm?

For credit, your solution needs to be polynomial in all parameters (n, m and k). Partial credit would be given for an efficient but suboptimal algorithm. The "fastest" algorithm seems to follow from using part (A).

[1] As fast as possible, you need to state the running time, explain the algorithm, provide pseudo-code if the description in words is ambiguous and unclear, explain why the algorithm is correct, etc.

[2] See first footnote.

[3] See other footnotes.