601.433/633 Introduction to Algorithms **Fall 2024**
Homework #6 **Due:** October 31, 2024, 9am

Remember: you may work in groups of up to three people, but must write up your solution entirely on your own. Solutions must by typeset (LaTeX preferred but not required). Collaboration is limited to discussing the problems – you may not look at, compare, reuse, etc. any text from anyone else in the class. Please include your list of collaborators on the first page of your submission. You may use the internet to look up formulas, definitions, etc., but may not simply look up the answers online.

Please include proofs with all of your answers, unless stated otherwise.

1 Reachable nodes (33 points)

Let G be a directed graph represented by an adjacency list. Suppose each node u has a weight $w(u)$, which might be different for each node. Give an algorithm that computes, for every node u, the maximum weight of any node that is reachable from u. So, for example, if G is strongly connected then every node can reach every other node, so for every node the maximum reachable weight is the same (the largest weight of any node in the graph).

More formally, for each vertex u let $R(u)$ denote the vertices reachable from u. Then when your algorithm is run on G, it should return an array of values where the value for node u is $\max_{v \in R(u)} w(v)$.

Your algorithm should run in $O(m+n)$ time. Prove correctness and running time.

2 Faster Shortest Paths (33 points)

Let $G = (V, E)$ be a directed graph with lengths $\ell : E \to \mathbb{R}$ with no negative-length cycles. Let $v \in V$, and let α denote the maximum, over all $u \in V$, of the number of edges on a shortest path from v to u (where the shortest-path is defined with respect to the lengths). Given G, ℓ, and v (but not α), give an algorithm that computes shortest path distances from v to all other nodes in $O(m\alpha)$ time. You may assume that $m = \Omega(n)$. Prove correctness and running time.

3 Traveling With a Coffee Constraint (34 points)

Let $G = (V, E)$ be a connected, undirected graph with positive edge lengths $\ell : E \to \mathbb{R}^+$. This graph represents a road network. You are trying to get from $s \in V$ to $t \in V$. Unfortunately, you are a coffee addict: getting caffeine often enough is crucial. There is a coffee shop at every vertex $v \in V$ (including s and t) where you can get coffee.

(a) (17 points) Since you care most about getting coffee regularly, what you really want to do is minimize the amount of time you are on the road between coffee stops. More formally, you want to know the minimum L so that there is a path from s to t in which every edge has length at most L. Design an $O(m \log n)$-time algorithm for this problem, and (as always) prove running time and correctness.

(b) (17 points) Now let's consider a variant where you're an addict and a snob: while you want coffee, you *hate* Starbucks. Unfortunately, you're even more of an addict than in the previous problem: there is some number $L \in \mathbb{R}^+$ so that if you go more that L distance without getting coffee, you will get a caffeine migraine and crash your car. So you simply *cannot* go more than L distance without stopping for coffee. So now you have the following problem: given a subset $S \subseteq V$ of vertices where the only coffee shop is a Starbucks, find the path from s to t which uses only edges of length at most L, and has the *fewest* vertices in S. You may assume that $s, t \notin S$. Design an $O(m + n \log n)$-time algorithm for this problem, and prove running time and correctness.