601.433/633 Introduction to Algorithms **Fall 2024**
Homework #8 **Due:** November 21, 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 Removing Train Stations (50 points)

We discussed in class how one of the original motivations for studying max-flow and min-cut comes from WWII, and in particular the questions of "how much stuff can be moved from s to t along the rail network" and "what rail lines can we remove in order to destroy the ability to ship anything from s to t". Now suppose that instead of being able to remove rail lines, we have the ability to sabotage rail *stations*. Can we still find the best way to do this?

More formally, suppose that we are given a directed graph $G = (V, E)$, two vertices $s, t \in V$ with $(s, t) \notin E$, and a function $c : V \setminus \{s, t\} \to \mathbb{R}^{\geq 0}$. As usual, $|E| = m$ and $|V| = n$. We will call $c(v)$ the *cost* of v. A feasible solution is a subset $S \subseteq V \setminus \{s, t\}$ so that there is no path from s to t in $G - S$ (the graph obtained from G by removing all vertices in S and all edges incident on at least one vertex in S). Your goal is to find the feasible solution S which has minimum cost, i.e., which minimizes $\sum_{v \in S} c(v)$.

Design an algorithm for this problem which runs in $O(mn)$ time. You may assume that $m \geq n$ and that there is an algorithm for maximum flow which runs in $O(mn)$ time (as we discussed in class: Orlin's algorithm). Prove running time and correctness.

2 Linear Programming (50 points)

Let's consider linear programming formulations of the minimum spanning tree problem. We now have an (undirected) graph $G = (V, E)$ and weights $w : E \to \mathbb{R}^+$. As we discussed in class, one way of phrasing the minimum spanning tree problem is as finding the minimum cost connected subgraph which spans all nodes. This interpretation naturally gives rise to a straightforward LP relaxation which requires every cut to have at least one edge crossing it (fractionally). More formally, suppose that we have a variable x_e for every edge e, and consider the following linear program. For all $S \subset V$, let $E(S, \bar{S})$ denote the edges with exactly one endpoint in S and exactly one endpoint not in S.

$$\min \sum_{e \in E} w(e) x_e$$

$$\text{subject to} \sum_{e \in E(S,\bar{S})} x_e \geq 1 \qquad \forall S \subset V : S \neq \emptyset$$

$$x_e \geq 0 \qquad \forall e \in E$$

Note that there are an exponential number of constraints, but let's not worry about that.

(a) (25 points) Prove that the optimal value of this LP is *at most* the weight of the minimum spanning tree.

(b) (25 points) Unlike the examples in class, this is not an exact formulation of the MST problem. Find a graph $G = (V, E)$ and weights $w : E \to \mathbb{R}^+$ such that the optimal LP value is strictly less than the weight of the MST (and prove this).