601.433/633 Introduction to Algorithms **Fall 2024**
Homework #7 **Due:** November 12, 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 Boruvka's Algorithm (40 points)

(a) (10 points) This was stated in class, but let's prove it formally. Let $G = (V, E)$ be an undirected graph and let $w : E \to \mathbb{R}_{\geq 0}$ be edge weights. Prove that if all edge weights are distinct ($w(e) \neq w(e')$ for all $e \neq e' \in E$) then there is a *unique* minimum spanning tree (MST).

Let $G = (V, E)$ be a connected, undirected graph and let $w : E \to \mathbb{R}_{\geq 0}$ be *distinct* edge weights (w is injective). We're going to analyze yet another MST algorithm: Boruvka's MST algorithm (from 1926), which is a bit like a distributed version of Kruskal.

We begin by having each vertex mark the lightest edge incident to it. (For instance, if the graph were a 4-cycle with edges of lengths 1, 3, 2, and 4 around the cycle, then two vertices would mark the "1" edge and the other two vertices will mark the "2" edge). This creates a forest F of marked edges. (Convince yourself that there won't be any cycles!). In the next step, each tree in F marks the shortest edge incident to it (the shortest edge having one endpoint in the tree and one endpoint not in the tree), creating a new forest F'. This process repeats until we have only one tree.

(b) (10 points) Show correctness of this algorithm by proving that the set of edges in the current forest is always contained in the MST.

(c) (10 points) Show how you can run each iteration of the algorithm in $O(m)$ time with just a few runs of DFS and no fancy data structures (heaps, union-find – remember, this algorithm was from 1926!). In other words, given a current forest F of marked edges, show how to find the set of edges which consists of the shortest edge incident to each tree in F in time $O(m)$.

(d) (10 points) Prove an upper bound of $O(m \log n)$ on the running time of this algorithm.

2 Clustering (30 points)

Consider the following problem. You are given a graph $G = (V, E)$ and a length function $\ell : E \to \mathbb{R}_{\geq 0}$. You should output a clustering (i.e., a partition) $(C_1, C_2, \ldots, C_k)$ of V into k clusters which maximizes the minimum distance between clusters. More formally, you should maximize $\min_{i,j \in [k], i \neq j} d(C_i, C_j)$, where $d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$ and where $d(x, y)$ is the shortest-path distance (under length function ℓ). Design an algorithm for this problem which runs in $O(m \log n)$ time, and prove correctness and running time.

Hint: Think about MSTs and Kruskal's algorithm

3 Matroids (30 points)

(a) (10 points) Let U be a finite set and let $k \geq 0$ be an integer. Let $\mathcal{I} = \{S \subseteq U : |S| \leq k\}$. Prove that $(U, \mathcal{I})$ is a matroid.

(b) (20 points) Let $(U, \mathcal{I})$ be a matroid, and let $w : U \to \mathbb{R}^+$ be an assignment of weights to elements. We know that the greedy algorithm will return a maximum-weight independent set. But suppose we are also given some $I \in \mathcal{I}$, and want to return the maximum-weight independent set *that contains I*. Modify the greedy algorithm to solve this problem, and prove that your algorithm is correct. You do not need to analyze running time, but your algorithm should need only a polynomial amount of time and calls to the independence oracle.