CS 535 Design and Analysis of Algorithms **Fall Semester, 2021**

Homework 6 version 1.1

Assigned: November 17 *Due: December 1*

Problem 1 The SET-COVERAGE problem has as input a collection of m sets $S_1, S_2, \cdots, S_m$, (given each as a list of distinct elements) and positive integers K and R, and asks if there exists K indices $1 \leq i_1 < i_2 < \cdots < i_K \leq m$ (called a *subcollection*) such that $| \cup_{j=1}^{K} S_{i_j}| \geq R$ (the union of the sets in the subcollection has size at least R).

MAXIMUM-SET-COVERAGE has as input a collection of m sets $S_1, S_2, \cdots, S_m$, (given each as a list of distinct elements) and positive integers K, and as objective find K indices $1 \leq i_1 < i_2 < \cdots < i_K \leq m$ such that $| \cup_{j=1}^{K} S_{i_j}|$ is maximized. In words, choose K sets such that they "cover" the maximum number of elements, an element being covered if it belongs to any of the chosen sets.

Assume there is a black-box polynomial-time algorithm for SET-COVERAGE, the decision version of MAXIMUM-SET-COVERAGE. That is, the algorithm can answer in polynomial-time if, given a collection of m finite sets $S_1, S_2, \cdots, S_m$, and positive integers K and R, there is a sub-collection with K sets such that their union has size at least R. Use this algorithm (note that you can use it repeatedly) to get a polynomial-time algorithm for MAXIMUM-SET-COVERAGE, or in other words, find a sub-collection with K sets whose union has the maximum size.

Since you are likely to use subroutines, makes sure you completely describe the input for each. Present pseudocode, analyze the running time, and argue correctness.

Problem 2 For a collection of m finite sets $S_1, S_2, \cdots, S_m$ (given each as a list of distinct elements), a *cover* is a subcollection of $S_1, S_2, \cdots, S_m$ such that the union of these sets equals the union of all sets (in which case we say that all the elements are covered).

Consider the following problem, called MINIMUM-SET-COVER: given a collection of m finite sets $S_1, S_2, \cdots, S_m$, find a cover with minimum number of sets.

SET-COVER is the decision version of MINIMUM-SET-COVER. In SET-COVER, a number q is given as part of the input in addition to the m sets, and the yes-no question is if there exists a cover with q sets. Prove that SET-COVER is NP-hard, using only the handout (precisely, the NP-hard problems given there).

MINIMUM-SET-COVER and MAXIMUM-SET-COVERAGE are related but not the same problem!.

Problem 3 Consider the following problem, called LONGEST SIMPLE $s-t$ PATH: Given a directed graph $G = (V, E, w)$, where $w(e)$ is defined as a non-negative integer for each edge $e \in E$, vertices $s, t \in V$, and a *positive* integer K, answer YES if there is a simple $s-t$ path of total weight at least K. An $s-t$ path starts at s and ends at t, a path is *simple* if it does not repeat any vertices, and the total weight of a path is the sum of the weights of its edges.

Prove that the LONGEST SIMPLE $s-t$ PATH problem is NP-hard. Simplest reduction is from HAMPATH.