CS 535 Design and Analysis of Algorithms **Spring Semester, 2019**

Homework 6 version 1.1

Assigned: April 18 *Due: May 6*

The late penalty is changed for this homework. In order to allow partial solutions to be posted before midterm, the latest time to submit (with penalty) is May 7.

Problem 1 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.

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.

Here is the original version: Show that, if $P = NP$, a polynomial time algorithm exists that, given a MINIMUM-SET-COVER instance, finds the cover with minimum number of sets. Note: NP is a class of *decision problems* or, if you prefer, *languages*, and producing a minimum-sized cover is a *function* of the input. Thus simply saying that ”because MINIMUM-SET-COVER is in NP, we are done” is not enough.

Replace the above paragraph by: Assume there is a black-box polynomial-time algorithm for the decision version of MINIMUM-SET-COVER. 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 a number k, there is a cover with k sets. Use this algorithm (note that you can use it repeatedly) to get a polynomial-time algorithm that finds a cover with minimum number of sets for any instance.

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 The SET-COVERAGE problem has as input a collection of m sets $S_1, S_2, \cdots, S_m$ and positive integers K and R, and asks if there exists K indices $1 \leq i_1 < i_2 < \cdots < i_K \leq m$ such that $|\cup_{j=1}^{K} S_{i_j}| \geq R$. Prove that SET-COVERAGE is NP-hard, using only the handout (precisely, the NP-hard problems given there).

Problem 3 Consider the following problem, called SHORTEST SIMPLE $s{-}t$ PATH: Given a directed graph $G = (V, E, w)$, where $w(e)$ is defined as a (possibly 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 most 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 SHORTEST SIMPLE $s - t$ PATH problem is NP-hard. Simplest reduction is from HAMPATH.