St. Francis Xavier University
Department of Computer Science

CSCI 550: Approximation Algorithms
Assignment 1
Due October 7, 2021 at 11:59pm (Atlantic time)

Assignment Regulations.

- This assignment must be completed individually. It is acceptable to discuss the assignment questions with other students, but you must write your answers individually and without assistance.
- Please include your full name and email address on your submission.
- You may either handwrite or typeset your submission. If your submission is handwritten, please ensure that the handwriting is neat and legible, and that the scanned paper is clear.

[10 marks] 1. Recall the weighted set cover problem from the lecture notes. Consider the *partial* weighted set cover problem:

> PARTIAL-WEIGHTED-SET-COVER
> Given: a ground set of elements $E = \{e_1, \ldots, e_n\}$, subsets $S_1, \ldots, S_m \subseteq E$, and a nonnegative weight $w_j \geq 0$ for each subset S_j
> Determine: the set $I \subseteq \{1, \ldots, m\}$ that minimizes $\sum_{i \in I} w_i$ such that, for some constant $0 < p < 1$,
>
> $$\left| \bigcup_{i \in I} S_i \right| \geq p|E|$$

Give a polynomial-time algorithm to find a solution to the partial weighted set cover problem whose value is no greater than $c(p) \cdot \text{OPT}$, where $c(p)$ is some constant depending on p and OPT is the value of the optimal solution to the weighted set cover problem (*not* the partial weighted set cover problem).

Hint. Adapt one of the algorithms we have for the weighted set cover problem, but modify the stopping condition appropriately. You may find the analysis in Section 1.6 of the course textbook to be useful.

[10 marks] 2. Given a graph $G = (V, E)$, we say that G contains an *independent set* if there exists a subset $S \subseteq V$ of vertices in the graph where, for any two vertices $u, v \in S$, the edge $\{u, v\}$ is not included in the edge set.

Given G as well as an integer k, we can define the independent set problem as follows:

> INDEPENDENT-SET
> Given: an undirected graph $G = (V, E)$ and an integer k
> Determine: whether G contains an independent set $S \subseteq V$ where $|S| \geq k$

Give a polynomial-time reduction from INDEPENDENT-SET to VERTEX-COVER.

Hint. Consider the relationship between an independent set and a vertex cover. If a graph G has one, then what can we conclude about G having the other?

[10 marks] 3. A *Steiner tree* is a tree that spans certain vertices in a graph $G = (V, E)$. In the node-weighted Steiner tree problem, we assume that each vertex in G has an associated weight and each edge in G has an associated cost, and we find a Steiner tree of least weight for G. The weight of a Steiner tree is the sum of the weights of each vertex and the costs of each edge in the tree.

Formally, we define the problem as follows:

NODE-WEIGHTED-STEINER-TREE
Given: an undirected graph $G = (V, E)$, a weight $w_v \geq 0$ for each $v \in V$, a cost $c_e \geq 0$ for each $e \in E$, and a set of terminal vertices $T \subseteq V$ Determine: a minimum-weight tree in G that spans all terminal vertices in T

Show that there exists no $c \ln(|T|)$-approximation algorithm for NODE-WEIGHTED-STEINER-TREE with $c < 1$, unless $\mathsf{P} = \mathsf{NP}$.

Hint. Use a reduction from another problem we discussed in the lecture notes.

[10 marks] 4. Consider the problem of scheduling on identical machines where each job to be scheduled is now subject to a *precedence constraint*. We say that $i \prec j$ if, in any feasible schedule, job i must be processed completely before job j can begin to be processed.

We also say that a job j is *available* if all jobs i such that $i \prec j$ have been processed completely. The list scheduling algorithm used by the machine is one in which, whenever a machine goes idle, any remaining job that is available is assigned to that machine to start processing.

Prove that this list scheduling algorithm is a 2-approximation algorithm for the identical-machine scheduling problem with predecence constraints.

Hint. Observe that Equation (2.4) in the course textbook gives us one lower bound for this variant of the identical-machine scheduling problem. If we have a set of jobs, all of which have precedence constraints, how might that give us another lower bound?