# Homework #1
# CS672 Spring 2020

Due Monday 2/10 at 2:10pm

**General Instructions** The following assignment is meant to be challenging. Feel free to discuss with fellow students, though please write up your solutions independently and acknowledge everyone you discussed the homework with on your writeup. I also expect that **you will not attempt to consult outside sources, on the Internet or otherwise,** for solutions to any of these homework problems. Finally, whenever a question asks you to "show" or "prove" a claim, please provide a formal mathematical proof.

**Note** In order to disincentivize skipping class to finish the homework, I have made it due within 10 minutes of the beginning of class. If you need more time, consider using one of your late days.

**Problem 1. (8 points; 4 points each part)**
W&S Problem 1.1

**Problem 2. (8 points; 4 points each part)**
W&S Problem 2.1

**Problem 3. (8 points)**
W&S Problem 2.3

**Problem 4. (8 points)**
In the weighted max-cut problem, you are given an undirected graph $G = (V, E)$, as well as an integer weight $w_e \geq 0$ for each edge $e \in E$. Denote $n = |V|$ and $m = |E|$, and let $b$ be the number of bits used to represent the weights. The goal is to find a cut $S \subseteq V$ maximizing the the total weight of edges with exactly one end in $S$. Recall that, in class, we presented a local search algorithm for this problem which runs in poly($n$) time and achieves an approximation ratio of $\frac{1}{2}$ when the graph is unweighted (i.e. $w_e = 1$ for all $e \in E$). The algorithm in question maintains a cut $S$ of the graph (initialized arbitrarily), and repeatedly adds or removes a vertex from $S$ so long such as such a local move increases the weight of the cut. When several improving local moves are possible, the algorithm chooses one arbitrarily.

**(a) [2 points].** Show that the local search algorithm described above runs in pseudopolynomial time, i.e. time polynomial in $n$ and $\max_e w_e$, and is a $\frac{1}{2}$-approximation algorithm for weighted max cut.

**(b) [4 points].** Modify the local search algorithm so that it takes a parameter $\epsilon > 0$, runs in time $\text{poly}(n, \frac{1}{\epsilon})$, and outputs a $(\frac{1}{2} - \epsilon)$-approximate solution to the weighted max cut problem.

**(c) [2 points].** Improve your result from (b) to a $\frac{1}{2}$-approximation algorithm which runs in polynomial time, i.e. time polynomial in $n$ and $b$.