Introduction to Algorithms	**Problem Set 7 (3 problems)**
CS 4820 Fall 2018	**Due 11:59pm Thursday, October 25**

Your homework submissions need to be typeset (hand-drawn figures are OK). See the course web page for suggestions on typing formulas.

The solution to each question need to be uploaded to CMS as a separate pdf file. To help provide anonymity in your grading, do not write your name on the homework (CMS will know it's your submission). (Questions with multiple parts need to be uploaded as a single file.) Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time. The running time must be bounded by a polynomial function of the input size.

(3 from last time) Scheduling Interviews II. In Problem 2 from problem set 6, we were assigning candidates to interviewers for a company without scheduling times for the interviews. In this problem, you are asked to set up a more detailed schedule. As in the previous problem, you are scheduling initial phone interviews for n job candidates at a company, with $k \leq n$ potential recruiters. Each candidate needs only one interview for now: concretely, each candidate needs to be matched to exactly one recruiter to be interviewed. The candidates are applying for different kinds of jobs, and recruiters are qualified to interview candidates only for some of the jobs. For each candidate i, you are given a list L_i of the recruiters who are qualified to interview that candidate. You also don't want to overload the recruiters, so each recruiter j has a limit q_j of the maximum number of interviews he or she can do.

Each interview will take one hour, and will be scheduled in one of T possible non-overlapping one-hour time slots $h_1, \ldots h_T$, each starting at the top of the hour (e.g. 10 AM, 1 PM, 5 PM ...). In addition to the input above, each interviewer and each candidate is asked to list the hours that they would be able to interview or be interviewed. Please note that each interviewer is asked to list strictly more than q_j possible slots to make scheduling easier.

Give an algorithm that finds an interview schedule if one exists. Let $m = \sum_i |L_i|$. Your algorithm should run in time polynomial in n, k, m, and T.

(2) Monitoring nodes in a graph. Consider a directed network $G = (V, E)$ that represents a transportation network in a war-torn region, where some edges of the network are not always reliable. The node $h \in V$ represents the location of the headquarters for an important strategic operation. In addition, there is a set of outposts or *terminals* $T \subseteq V$ that the headquarters will need to send supplies to. You want to know how vulnerable the operation is to disruption. For each terminal $v \in T$, you have a value $w_v \geq 0$ of the importance of being able to reach v from the headquarters. You wonder how easily the enemy could damage the supply network by disconnecting a few specific edges.

We make this precise as follows. Deleting a set of edges $F \subseteq E$, some nodes in T will no longer be reachable from h, i.e., these are the nodes such that there is no h-v path in the subgraph $(V, E - F)$. We will call such terminals *disconnected by* F, and let $q(F)$ denote the total value of the terminals $v \in T$ disconnected by F. In other words, $q(F)$ is value unreachable terminals in T when losing access to the edges F.

Give a polynomial-time algorithm to find a set F of edges that maximizes the quantity $q(F) - |F|$. (Note that setting F equal to the empty set is an option, so this maximum will be nonnegative.) Update 10/20: you may assume that the values w_v are non-negative integers, and your algorithm may also be polynomial in the values of the terminals.

(3) Reduction. Suppose you have a subway map that you can represent as an undirected graph $G = (V, E)$ of stops, with n different subway lines described as paths through the graph. Assume no more than one train is scheduled to be at a stop at a time. (An edge can belong to one or more subway lines.)

One day a week, you want someone to quickly clean the front window of every train. To do this, you can hire somebody to stay at a particular train stop $v \in V$ and every time a train is at that stop, they will clean the front window of it, but will not board. You want to ensure that every train will, at some point, be at a stop with one of those people. However, you only have the budget to hire k window cleaners to occupy k distinct stops. You would like to find out if it is possible to use only k window cleaners to reach all the trains. We call this the WindowCleaner Problem.

Show that the WindowCleaner Problem is at least as hard as one of the two hard problems we considered in class on Friday, Oct 19th: the Vertex Cover problem or the Independent Set problem. (This coming week, we'll show that both of these are NP-complete.)