Introduction to Algorithms	**Problem Set 2**
CS 4820 Spring 2016	**Due 11:59pm Thursday, February 11**

Your homework submissions need to be typeset (hand-drawn figures are OK). See the course web page for suggestions on typing formulas. Solution to each question needs to be uploaded to CMS as a separate pdf file. (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.

(1) The manager of a large student union on campus comes to you with the following problem. She's in charge of a group of n students, each of whom is scheduled to work one *shift* during the week. There are different jobs associated with these shifts (tending the main desk, helping with package delivery, rebooting cranky information kiosks, ...) but we can view each shift as a single contiguous interval of time. There can be multiple shifts going on at once.

She's trying to choose a subset of these n students to form a *supervising committee* that she can meet with once a week. She considers such a committee to be *complete* if, for every student not on the committee, their shift overlaps (at least partially) the shift of some student who is on the committee. In this way, each student has an opportunity during his or her shift to report any issues to someone who's serving on the committee.

Give an efficient algorithm that takes the schedule of n shifts and produces a complete supervising committee containing as few students as possible.

Example: Suppose $n = 3$, and the shifts are

Student 1: Monday 4 PM - Monday 8 PM,
Student 2: Monday 6 PM - Monday 10 PM,
Student 3: Monday 8 PM - Monday 11 PM.

Then the smallest complete supervising committee would consist of just the second student, since the second shift overlaps both the first and the third.

(2) Your friends are running a Xeroxing service. Due to maintenance problems they are down to a single machine. Early morning as the shop opens they have n customers who bring in a Xeroxing job each. Customer i's job will take $t_i > 0$ time to complete. Your friends wonder what is the best order to do these jobs. Given a schedule (i.e., and ordering of the jobs), let C_i denote the finishing time of job i. For example, if job j is done right after job i we would get $C_j = C_i + t_j$. Given the maintenance problem, your friends worry that the customers will get upset by long waits. They decided that a customer's unhappiness with their service is directly proportional to the finishing time of his/her job, and they assign a weight to each customer representing the priority of the customer, i.e. how important is their happiness for the business. They would like to order the jobs so as to minimize the weighted total of the unhappiness, i.e., to minimize $\sum_{i=1}^{n} w_i C_i$.

They are considering three different greedy algorithms:

(a) order jobs in increasing order of the time their require, i.e., do short jobs first.

(b) order jobs in decreasing order of weight, i.e., jobs for important customers first.

(c) order jobs in decreasing order of the w_i/t_i ratios.

Assume for simplicity that all jobs require different amount of time, all customers have different weights, and all ratios w_i/t_i are different.

For each of these proposed greedy algorithms, either prove that the algorithm is guaranteed to produce the optimal schedule (minimizing the total $\sum_{i=1}^{n} w_i C_i$), or give an example to show that this is not the case.

(3) A group of network managers are considering improving roads in a large network. The road network they consider is represented by an undirected graph $G = (V, E)$ with n nodes and m edges, where each node is a city, and each edge e is a road. They are also given estimated costs c_e for upgrading each road e. Assume that all edge costs are different. They want to upgrade roads along a minimum cost spanning tree, so that any two locations will be connected via improved edges. Before planning even begins, they learn about the state of one edge e, and get a number of requests from some special interest groups, to include certain edges in the upgrading project. However, they cannot afford to increase the total cost.

To help them respond to such a request fast, give an $O(m + n)$ algorithm for the following decision problem: given a graph $G = (V, E)$ with n nodes and m edges, costs on the edges, and a selected edge e, is edge e contained in the minimum spanning tree T of the graph G? Note that we do not know an $O(m + n)$ implementation of either of the Minimum Spanning Tree algorithms, what you are asked to do is decide if a given edge e is in the MST in $O(m + n)$ time.[1]

[1]You may be curious to know that one can also solve the bit harder problem in $O(m + n)$ time: given an edge e how much does the cost of the minimum cost spanning tree increase if we have to include edge e.