CSCI 570 (Practice) Final Exam Summer 2016

Last (family) Name	
Rest of Name	
ID	

DO NOT OPEN EXAM UNTIL INSTRUCTED TO DO SO

SILENCE MOBILE PHONE AND OTHER DEVICES

PLEASE READ THE COVER CAREFULLY

- You are allowed to have one normal-sized page of hand-written notes.
- You may not have any other materials available to you other than those used for writing.
- You will have 120 minutes, from **2:30 PM** until **4:30 PM**, to work on this exam.
- When answering any given question, use only the page on which the question is printed.
- If you leave your seat without permission of a proctor, you may be required to submit your exam without any further changes.
- If you give multiple answers to a free-response question and do not clearly indicate which one you wish to be graded, we reserve the right to select which one to grade.
- Please keep this cover page and the question pages intact.
  (You may remove the scratch paper from the end if you would like)

Question	Points	Possible
1		10
2		10
3		10
4		10
Total		40

1. When we studied greedy algorithms in lecture, we saw a scheduling problem with deadlines, in which we sought to minimize the maximum lateness. In this problem, we were given a set of n jobs, each with a deadline d_i and a processing time t_i, and had to assign each job a starting time s_i in such a way that no two jobs overlapped.

   Suppose instead of allowing a job to be finished after its deadline, a job became worthless if it wasn't completed in time; that is, if we assigned a job s_i such that $s_i + t_i > d_i$, we might as well not have included it at all. Our goal now is to select a set of jobs J and a start time for each of these jobs such that all get done by their deadline, and to make this set J as large as possible.

   Give a **dynamic programming** algorithm to find the largest set J, assuming that all deadlines d_i and running times t_i are integers. Your algorithm should by polynomial in the number of jobs in the input n and the value of the maximum deadline $D = \max_i d_i$. $O(nD)$ time is possible, but full credit will be given for $O(n^2D)$

   Provide the recursive solution (including the base case), explain *briefly* why it is correct, and state what the running time of the iterative algorithm would be.
   You *do not* need to give the iterative algorithm.

   Hint: For any optimal set J, the jobs within J are scheduled earliest deadline first. You may assume the jobs are given already sorted by deadline if you would like.

2. Let X be a set of n intervals on the real line. We say that a set P of points *collects* X if every interval in X contains at least one point in P. Give an efficient **greedy** algorithm that takes as input a set X of intervals and finds the smallest collecting set. Prove that your algorithm correctly minimizes the size of this set.

   (a) A correct algorithm to solve this involves selecting one point in a particular manner, adding it to your set, removing every interval that includes this point, and recurse on the remaining intervals.

   Describe how you would select the first point to add to your set in such a way that the previous description will produce the optimal representative set.

   Your answer to this part may be at most 30 words.

   (b) Prove that your algorithm minimizes the size of the representative set.

3. Suppose I am teaching a large class and have many TAs. I want a total of H office hours to be held each week, and I have a set of k hour-long time intervals $I_1, I_2, \ldots I_k$ in which the TA office hours room is available. I have collected, from each TA, a list of which subset of the time intervals he or she can hold office hours. I also know, for each TA, the maximum m_j hours he or she can hold office hours, and I have for each a minimum l_j hours he or she should be required to hold. Lastly, I have some value H for the total number of hours I'd like held during the week.

   Use **network flow techniques** to design a polynomial-time algorithm that takes as input an instance of this problem (the time slots, the TA schedules, the sets of l_j and m_j values, and the value H) and determines if there is a valid way to schedule the TA's office hours to respect these constraints.

   (For this question and questions like it, you should clearly indicate how you would build a graph, how large the graph is, what a unit of flow means on this graph, and what you are looking to find from the network flow output.)

   Hint: If you are struggling to start this problem, imagine that there are no lower bounds on the TA's hours

4. The LOAD BALANCING problem is as follows. We are given a set of m machines and a set of n jobs. Job j takes processing time t_j. Our goal is to determine if it is possible if we can assign each job to a machine so that each machine takes the same total time to process its set of tasks.

   Prove that LOAD BALANCING is $\mathcal{NP}$-complete.

Extra page. You may use this for scratch paper, but nothing on this page will be graded.