CSCI 570, Summer 2016 Homework 4

For each of the following problems, you will use network flow. Each requires polynomial time, but does not require a more specific running time. You may (and should!) use "maximum flow" (or minimum cut) as a sub-routine without needing to explain it or its running time; when grading, we will assume that you call a polynomial time solution to network flow. Note that the graph you provide must still be polynomial in size.

For each question, explain how to construct the graph that will be used as input for the network flow algorithm, why that graph is polynomial in size, and how you will interpret the output of that algorithm to solve the problem.

This homework due date has been moved to **August 3** so you can focus on the quiz. Just the same, I encourage you to start this early.

1. Suppose we are given a set of tasks S where each task $i \in S$ has an *arrival time* a_i, a *deadline* d_i, and a *workload* w_i. For each unit of time, we can do one unit of workload for the task, and we do not need to do it continuously (we can stop a task and resume it later). Our goal is to determine if we can complete all tasks by their deadline.

   (a) Use network flow techniques to design a polynomial-time algorithm that will determine if the tasks can all be done by their deadlines. Explain why your algorithm is polynomial time.

   (b) An **overworked period** is a pair of times A and B with the following property. If we let S_{AB} be the set of tasks whose arrival time is A or later and deadline is B or earlier (i.e. $A \leq a_i \leq d_i \leq B$) then the total workload exceeds the time (i.e. $\Sigma_{i \in S_{AB}} w_i > B - A$). If there is an overworked period, then there is no way to complete all the tasks by their deadlines.
   Use network flow techniques to design a polynomial time algorithm that determines if there is an overworked period, and if so, what it is.

2. Suppose we'd like to acquire a set of n useful programs. Each program is sold by two companies, and the two versions are potentially of different quality. Suppose that company one's version of program i has quality x_i and company two's version has quality y_i. The obvious thing to do is to get the higher quality version for every program, but unfortunately the programs need to be *compatible* with one another and using incompatible versions (for example using company one's word processor and company two's web page editor) causes some amount of annoyance. Suppose we can evaluate this annoyance at $a_{(i,j)}$ for using company one's version of program i and company two's version of program j. Our goal is to select sets S_1 and S_2 of programs to buy from company one and two respectively, such that $S_1 \cup S_2 = \{1, 2, ..., n\}$, $S_1 \cap S_2 = \emptyset$, and we maximize:

$$\sum_{i \in S_1} x_i + \sum_{j \in S_2} y_j - \sum_{i \in S_1, j \in S_2} a_{(i,j)}$$

   Use network flow techniques to design a polynomial time algorithm for this problem.

   Hint: You can do this with only $n + 2$ vertices. This is not a matching problem.

   Hint 2: Maximizing that quantity is equivalent to minimizing some other quantity.

What? Only two problems? If you would like additional practice, the following questions are suggested. Do not submit them for credit; these are for your own practice.

G & T: A-16.1 through A-16.7.

K & T: Chapter 7, problems 1, 2, 3, 4, 10, 23, 24 (if you struggle on 23, try doing 24 first). For practice with using network flow, try Chapter 7, problems 8, 9, 11, 13, 14, 16, 17, 18, 21, 23, 24, 51.