Homework 7

Winston (Hanting) Zhang

Friday, November 4, by 23:00

General Instructions. The following assignment is meant to be challenging, and we anticipate that it will take most of you at least 10–15 hours to complete, so please allow yourself plenty of time to work on it. We highly recommend reading the entire assignment right away — you never know when inspiration will strike. Please provide a formal mathematical proof for all your claims, and present runtime guarantees for your algorithms using asymptotic (big-$O/\Omega/\Theta$) notation, unless stated otherwise. You may assume that all basic arithmetic operations (multiplication, subtraction, division, comparison, etc.) take constant time.

Collaboration. Please carefully check the collaboration policy on the course website. When in doubt, ask an instructor.

Consulting outside sources. Please carefully check the policy regarding outside sources on the course website. Again, when in doubt, ask an instructor.

Submission. Homework submission will be through the Gradescope system. Instructions and links have been provided through the course website and Piazza. The only accepted format is PDF. Starting with this homework, only typed solutions will be accepted, and we highly recommend producing your solutions using LaTeX (the text markup language we are also using for this assignment).

Recommended practice problems (do not hand in): KT Problems 7.6, 7.7, 7.9, 7.12, 7.13, 7.16, 7.19, 7.22, 7.23, 7.25, 7.27, 7.35

Problem 1. [15]

When we defined flows in class, we did not rule out that there might be cycles in the flow. Having cycles in your flow seems kind of pointless at best, and a nuisance at worst. Here, you are going to prove a formal version of this intuition, by designing an algorithm that proves that you do not need cycles to get good flows. To be precise, when we say that a flow f "contains cycles", what we mean is that the set of edges with positive flow, i.e., $\{e \in E\} f_e > 0$, contains a cycle.

Your algorithm will be given a graph $G = (V, E)$ with non-negative edge capacities c_e, a source s, sink t, and a valid flow f. Also, there will be no edge into s or out of t; otherwise, the definition of the value of a flow is a bit strange. It is supposed to run in polynomial time (not pseudo-polynomial) and output a new flow f' with $f'_e \leq f_e$ for all edges e (so you cannot add flow to any edges), of the same value $\nu(f') = \nu(f)$, and such that f' is acyclic.

Give and analyze a polynomial-time algorithm for finding such an f'.

Problem 2. [15]

Imagine that you are starting a food delivery startup with the motto "We know you hate when you must wait. Herewith we state that it's not great when we are late, so it's our fate to not get paid.[1]" The business model is as follows: you have m delivery drivers. Customers place food orders from restaurants of their choosing, and state a deadline of how much maximum they are willing to wait for their food. They pay the restaurant for the food, and possibly you for delivery. If your company accepts an order from a customer, you promise to deliver it within the specified time limit; otherwise, the (flat) delivery fee of \$10 is waived.[2] Your high-level goal is now to select as many orders to accept as possible (and thus to make as much money as possible), but subject to not being late on any of them.

To be more precise, you are choosing from among n potential customers. For each customer i, you are told where they live (let's call that location L_i), and where the restaurant is that they are ordering from (let's call that R_i). You are also told how many minutes they are willing to wait; this is a number $t_i > 0$. In addition, you know the locations of your m delivery drivers (let's call the location of the j-th driver D_j).

Each delivery driver can only be assigned at most one order (even though some orders may be close together, or close to where the driver lives). If a driver is in charge of an order, they first drive from their location to the restaurant, where they pick up the food. You can assume that picking up food always takes exactly one minute. From the restaurant, they then drive to the customer's house. You have a route planning software (think Google Maps) into which you or your program can enter any pair of locations (restaurants, customer locations, driver locations) and get a precise (and always completely accurate) answer of how long it will take to drive from the first location to the second. Remember that for each order that you serve on time, you get \$10, and for any other order (which you decline or for which you are late), you get nothing.

Give and analyze a polynomial-time algorithm for maximizing the profit that your company makes. Your algorithm should output both the profit and the assignment of drivers to orders that gives that profit.

Problem 0.

Chocolate Problem: 1 chocolate bar

Problem 7.51 from the textbook.

[1]There wasn't enough budget to hire someone to come up with a good marketing slogan.

[2]The customer still pays for the food, but since that money goes directly to the restaurant, it doesn't help you.