CS 577: Introduction to Algorithms Fall 2019

Homework 8

Instructor: Dieter van Melkebeek TA: Nicollas Mocelin Sdroievski

This homework covers network flow applications. In order to get feedback on problem 3, you need to submit it via Canvas by **11:59pm on 11/12**. Please refer to the homework guidelines on Canvas for detailed instructions.

Warm-up problems

1. Your friends are involved in a large-scale atmospheric science experiment. They need to get good measurements on a set S of n different conditions in the atmosphere (such as the ozone level at various places), and they have a set of m balloons that they plan to send up to make these measurements. Each balloon can make at most two measurements.

   Unfortunately, not all balloons are capable of measuring all conditions, so for each balloon $i = 1, 2, \ldots, m$, they have a set S_i of conditions that balloon i can measure. Finally, to make the results more reliable, they plan to take each measurement from at least k different balloons. (Note that a single balloon should not measure the same condition twice.) They are having trouble figuring out which conditions to measure on which balloon.

   (a) Design a polynomial-time algorithm that takes the input to an instance of this problem (the n conditions, the sets S_i for each of the m balloons, and the parameter k), and decides whether there is a way to measure each condition by k different balloons, while each balloon measures at most two conditions.

   (b) You show your friends a solution computed by your algorithm from (a), and to your surprise they reply, "This won't do at all – one of the conditions is only being measured by balloons from a single subcontractor." You hadn't heard anything about subcontractors before; it turns out there's an extra wrinkle they forgot to mention. Each of the balloons is produced by one of three different subcontractors involved in the experiment. A requirement of the experiment is that there be no condition for which all k measurements come from balloons produced by a single subcontractor. Explain how to modify your polynomial-time algorithm from part (a) into a new algorithm that decides whether there exists a solution satisfying all the conditions from (a), plus the new requirements about sub-contractors.

2. The Packers are playing the Bears tonight, and you'd like to invite some of your friends to watch the game at your place. All of your friends love football, and are either Packers or Bears fans, but some of them are known not to get along very well. In order to avoid possible trouble, you do not want to invite two people who are on bad terms with each other *and* root for a different team. (Having people who are on bad terms but root for the same team is OK, as is any of the other two combinations.) Also, although you like all of your friends, you like some better than others, and you have assigned a positive value to each of your friends.

   Design a polynomial-time algorithm to figure out which friends to invite so as to maximize their total value under the above constraints.

Feedback problem

3. You are building a system consisting of n components. There are two possible suppliers for each component: Alpha and Omega. Alpha charges α_i for component i, and Omega charges ω_i. You'd like to spend as little money as possible, but also want to take the costs due to incompatibilities between components of different suppliers into account. In particular, if you buy components i and j from different suppliers, there is an incompatibility cost of $c(i, j)$.

   Design an efficient algorithm to determine from which supplier you should buy the components so as to minimize the sum of the purchase costs and the incompatibility costs.

Additional problem

4. Some of your friends with jobs out West decide they really need some extra time each day to sit in front of their laptops, and the morning commute from Woodside to Palo Alto seems like the only option. So they decide to carpool to work.

   Unfortunately, they all hate to drive, so they want to make sure that any carpool arrangement they agree upon is fair, and doesn't overload any individual with too much driving. Some sort of simple round-robin scheme is out, because none of them goes to work every day, and so the subset of them in the car varies from day to day.

   Here's one way to define fairness. Let the people be labeled $S = \{p_1, \ldots, p_k\}$. We say that the *total driving obligation* of p_j over a set of days is the expected number of times that p_j would have driven, had a driver been chosen uniformly at random from among the people going to work each day. More concretely, suppose the carpool plan lasts for d days, and on the ith day a subset $S_i \subseteq S$ of the people go to work. Then the above definition of the total driving obligation Δ_j for p_j can be written as $\Delta_j = \sum_{i:p_j \in S_i} \frac{1}{|S_i|}$. Ideally, we'd like to require that p_j drives at most Δ_j times; however, Δ_j may not be an integer.

   So let's say that a driving schedule is a choice of a driver for each day -— i.e., a sequence $p_{i_1}, p_{i_2}, \ldots, p_{i_d}$ with $p_{i_t} \in S_t$ -— and that a fair driving schedule is one in which each p_j is chosen as the driver on at most $\lceil \Delta_j \rceil$ days.

   (a) Prove that for any sequence of sets $S_1, \ldots, S_d$, there exists a fair driving schedule.

   (b) Design an algorithm to compute a fair driving schedule in time polynomial in k and d.

Challenge problem

5. Here is a variant of the game "Six Degrees of Kevin Bacon."

   You start with a set X of n actresses and a set Y of n actors, and two players P_0 and P_1. Player P_0 names an actress $x_1 \in X$, player P_1 names an actor y_1 who has appeared in a movie with x_1, player P_0 names an actress x_2 who has appeared in a movie with y_1, and so on. Thus, P_0 and P_1 collectively generate a sequence $x_1, y_1, x_2, y_2, \ldots$ such that each actor/actress in the sequence has costarred with the actress/actor immediately preceding. A player $P_i (i = 0, 1)$ loses when it is P_i's turn to move, and she cannot name a member of her set who hasn't been named before.

   Suppose you are given a specific pair of such sets X and Y, with complete information on who has appeared in a movie with whom. A *strategy* for P_i in our setting is an algorithm that takes a current sequence $x_1, y_1, x_2, y_2, \dots$ and generates a legal next move for P_i (assuming it's P_i's turn to move). Design a polynomial-time algorithm that, given some instance of the game, decides at the start of the the game which of the two players can force a win.

Programming problem

6. SPOJ problem Bank robbery (problem code BANKROB).