Problem Set 8: Flow Applications

1. (Easy) Network flow issues come up in dealing with natural disasters and other crises, since major unexpected events often require the movement and evacuation of large numbers of people in a short amount of time.

   Consider the following scenario. Due to large scale flooding in a region, paramedics have identified a set of n injured people distributed across the region who need to be rushed to hospitals. There are k hospitals in the region, and each of the n people needs to be brought to a hospital that is within a half-hour's driving time of their current location (so different people will have different options for hospitals, depending on where they are right now).

   At the same time, one doesn't want to overload any one of the hospitals by sending too many patients its way. The paramedics are in touch by cell phone, and they want to collectively work out whether they can choose a hospital for each of the injured people in such a way that the load on the hospitals is balanced: Each hospital receives at most $\frac{n}{k}$ people.

   Give a polynomial-time algorithm that takes the given information about the people's locations and determines whether this is possible.

2. (Medium) Suppose in a directed graph G, there are k edge-disjoint paths from s to t and from t to u. Are there k edge-disjoint paths from s to u?

3. (Medium) You are running a company with n employees. At the beginning of the year, each employee has specified a subset of days during the year during which he or she is not available. You would like to ensure that on every day at least ℓ employees report to work, and no employee comes to work for more than x days during the year. Show how you can solve this problem efficiently.

4. (Medium) Several families go out to dinner together. To increase their social interaction, they would like to sit at tables so that no two members of the same family are at the same table. Show how to formulate finding a seating arrangement that meets this objective as a maximum flow problem. Assume that the dinner contingent has p families and that the ith family has $a(i)$ members. Also assume that q tables are available and that the j-th table has a seating capacity of $b(j)$.

5. (Medium) The Miso Good food truck produces a large variety of different lunch menu items. Unfortunately, they can only produce their foods in limited quantities, so they often run out of popular items, making customers sad.

   To minimize sadness, Miso Good is implementing a sophisticated lunch-ordering system. Customers text in their acceptable choices before lunch time, and the truck can use an algorithm to preassign lunches to customers. Customers who do not get one of their choices should receive a \$10 voucher. Miso Good would like to minimize the number of vouchers they give out.

   Give an efficient algorithm for Miso Good to assign lunches to customers. In general, suppose that on a given day, Miso Good has produced m types of food items $b_1, \ldots, b_m$, and the quantity of each type of food item b_j is exactly q_j. Suppose that n customers $a_1, \ldots, a_n$ text in their preferences, where each customer a_i submits a set A_i of one or more acceptable lunch choices. The algorithm should assign each customer either one of his/her choices or a \$10 voucher, while minimizing the number of vouchers distributed.

6. (Medium) Your friends are attending a convention and want to ask questions to the panelists. They worked together to create a set S of n different topics they would like to ask about. Each of your m friends has a VIP pass that guarantees them the ability to ask at most three questions each.

   Unfortunately, due to the dense subject matter of the convention, not all of your friends understand every topic well enough to ask about it. For each friend $i = 1, 2, \ldots, m$, they have a set S_i of topics they are capable of asking about. Finally, to make sure each topic is thoroughly addressed, the friends want to ask at least k different questions about each of the n topics. (Note that one person should not ask about the same topic more than once.) They are having trouble figuring out who should ask about which topics.

Design a polynomial-time algorithm that takes the input to an instance of this problem (the n topics, the sets S_i for each of the m friends, and the parameter k) and decides whether there is a way to ask k questions about each of the n topics, while each friend asks at most three questions each.

7. (MEDIUM) You have invited n friends to a party. There are k tables in your home, and each table can accommodate l people. There are s schools in the neighbourhood, and each of your friends attends one of these schools. You would like to ensure that at most 2 people from the same school are seated at the same table. Show how you can use maximum flow formulation to find such a seating arrangement (or declare that no such arrangement is possible).

8. (HARD) You are given an $n \times n$ matrix X with real positive entries. You would like to round each entry X_{ij} in the matrix to either $\lfloor X_{ij} \rfloor$ or $\lceil X_{ij} \rceil$ such that the row sums or column sums do not change (i.e., for any row, the sum of the rounded entries is equal to the sum of the actual X_{ij} values in that row, and similarly for any column). Assume that you are given a matrix where such a rounding is possible

   Show how you can solve this problem efficiently. (Hint: Use circulation with demand and lower bound)

$$\begin{array}{ccc} 1.2 & 3.4 & 2.4 \\ 3.9 & 4.0 & 2.1 \\ 7.9 & 1.6 & 0.5 \end{array} \longrightarrow \begin{array}{ccc} 1 & 4 & 2 \\ 4 & 4 & 2 \\ 8 & 1 & 1 \end{array}$$

9. (HARD) Imagine a group of robots trapped in an $n \times n$ grid. Each robot starts from a unique position in the grid, called a *terminal*. The grid has boundary edges that lead to safety, and each robot must reach an exit on the boundary without overlapping paths – no two robots can occupy the same cell on their paths.

   Each cell in the grid connects to its four adjacent cells (left, right, top, and bottom), except for boundary cells. The goal is to determine if it's possible to find separate paths from each terminal to a distinct boundary cell, ensuring each robot can escape without path conflicts.

   Design and analyze an efficient algorithm to find these vertex-disjoint paths, if they exist, enabling a safe escape for all robots.