CS 577: Introduction to Algorithms — Homework 9

Out: 04/19/18 — **Due: 04/26/18**

Ground Rules

- The homework is worth 5 points (out of a total of 100 points you can accumulate in this course).
- The homework is to be done and submitted in pairs. You can partner with someone from either section.
- The homework is due at the beginning of either lecture on the due date. No extensions to the due date will be given under any circumstances.
- Turn in your solution to each problem on a **separate sheet** of paper (or sheets stapled together), with your names clearly written on top.

Note

- When we ask for an efficient algorithm, it means that the running time for the algorithm should be some polynomial in the size of the problem. For graph problems, the size is $|V| + |E| = n + m = O(n^2)$, so we require an algorithm with running time $O(n^c)$ for some constant c.
- When we ask you to analyze an algorithm, provide a proof of correctness as well as an analysis of running time.

Problems

1. **(Worth: 3 points. Page limit: 1 sheet; 2 sides)** A group of people are carpooling to work together, and want to come up with a fair schedule for who will drive on each day. This is complicated by the fact that the subset of people in the car varies from day to day due to different work schedules. We define fairness as follows. There are n people and d days. Let $S = \{1, \cdots, n\}$ denote the set of people. Suppose that on the i-th day a (nonempty) subset $S_i \subseteq S$ go to work. A schedule for the d days is an assignment of drivers to days. The driver for day i must belong to the subset S_i. Note that a schedule may be partial in that not all days get assigned a driver. A schedule is called "fair" if for each driver j, the total number of times they drive is at most

$$\Delta_j = \left\lceil \sum_{i: j \in S_i} \frac{1}{|S_i|} \right\rceil .$$

   (a) Design and analyze an efficient algorithm for computing a fair schedule that maximizes the number of days to which a driver is assigned.

   (b) Prove that for any choice of sets $S_1, ..., S_d$, there exists a fair schedule that assigns a driver to *each* day.

   Hint: For the first part, reduce the problem to network flow. What can you say about the value of the maximum flow in your constructed network?

(Turn over for Problem 2)

2. **(Worth: 2 points. Page limit: 1 sheet; 2 sides)** The manager at a local toy store, Algos-R-Us, is in need of some algorithmic expertise. A few days back he received a shipment of Russian nesting dolls that was damaged in transit. All of the sets were disassembled, that is, none of the dolls were nested inside another. There are n dolls in all and k boxes to pack them into. The manager needs to figure out how to assemble the n dolls into k or fewer nested sets, if at all possible. For any two dolls, the only way to tell whether one can be nested inside the other is by direct examination. For some pairs of dolls, neither can be nested inside the other (e.g., due to one being shorter but wider than the other). Design and analyze an efficient algorithm to find an arrangement of the n dolls into k or fewer nested sets, if such a partition exists.

   To be precise, the input to your algorithm consists of the numbers n and k, and for each pair $i, j \in [n]$, a bit that specifies whether or not doll i can be nested inside doll j. Your algorithm should return an arrangement of the n dolls into k or fewer nested sets, if such an arrangement exists, and return "Error" if not.

   Hint: Think of the arrangement of dolls as a matching that matches each contained doll with the doll that directly contains it. So each doll is matched at most once to a doll containing it, and at most once to a doll contained in it.