# CS 577: Introduction to Algorithms — Homework 9

**Out: 04/17/17** — **Due: 04/24/17**

## Ground Rules

- Review problems will be discussed at the following week's review.
- Self-graded problems will be discussed at the following week's review, and are to be self-graded by you during the review. Only students present at the review session and turning in their graded solutions in person will earn points.
- Graded problems are to be turned in on the due date at the beginning of the lecture.
- Self-graded problems should be done individually.
- Graded problems should be done in pairs.
- Please follow page limits for all self-graded and graded problems.
- Write your name(s) clearly on your submissions, and turn in each problem on **a separate sheet of paper**.

## Review Problems

1. Suppose you work for Facebook and you want to find groups of users which are "close-knit". Informally, you have a parameter $\alpha$ in mind, and you want to find a group of users $S$ such that the number of friends among the users in $S$, call this $f(S)$, is at least an $\alpha$ fraction of the total number of users in the set. Formally, you're given a graph $G = (V, E)$ and a parameter $\alpha$ between 0 and 1; for any $S \subseteq V$, define $f(S)$ to be the number of edges in $E$ with both endpoints in $S$. Give an efficient algorithm which determines whether there exists a set $S \in V$ such that $f(S)/|S| \geq \alpha$.

2. Suppose we are given an array $A[1 \cdots m][1 \cdots n]$ of non-negative real numbers. We want to round $A$ to an integer matrix, by replacing each entry $x$ in $A$ with either $\lfloor x \rfloor$ or $\lceil x \rceil$, without changing the sum of entries in any row or column of $A$. For example:

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

   Describe and analyze an efficient algorithm that either rounds $A$ in this fashion, or reports correctly that no such rounding is possible.

## Self-graded Problems

3. **(Page limit: 1 sheet; 2 sides)** Two companies Pineapple and Nanosoft make competing versions of $m$ different software products. Pineapple charges $p_i$ for product $i$ and Nanosoft charges $n_i$ for its version of $i$. A consumer wants to buy one version of each product. While the customer prefers cheaper versions, she also prefers to buy most software from the same company. In particular, products $i$ and $j$, if bought from different companies, impose an incompatibility cost of $c(i, j)$ on the customer. The customer's total cost from buying software is the prices paid to the two companies, plus a sum over all incompatible pairs of the respective incompatibility costs. The goal is to determine, for each software product, from which company the customer should buy that product so as to minimize her total cost.

   Design and analyze an efficient algorithm for this problem.

## Graded Problems

4. **(Page limit: 1 sheet; 2 sides)** (Taken from "Algorithm Design") 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. Let $S = \{1, \cdots, n\}$ denote set of people. Suppose on the $i$-th day a subset $S_i \subseteq S$ go to work. A schedule for $d$ days is a sequence $i_1, i_2, \cdots, i_d$ specifying the driver for each day. A schedule is 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 (potentially partial) fair driving schedule that maximizes the total 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 driving schedule that assigns a driver to *every* 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?*

5. **(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, it is possible to tell if one can be nested inside the other, but there is no other way of telling whether two dolls belong to the same set. For some pairs of dolls, neither can be nested inside the other (e.g. one may be taller than the other and the other wider than the first). 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.*