CS41 Lab 1

January 22, 2015

In typical labs this semester, you'll be working on a number of problems. You are encouraged to work with one or two others. You will not be handinging in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design. However, it will be common to have some overlap between lab exercises and homework sets.

1. **Harry's Hoagie House.** Harry has a hoagie house, famous for its special-order hoagies. These hoagies cost \$5, no matter what ingredients are used or how long it takes to prepare the hoagies. Needless to say, this deal attracts a lot of regular customers, some of whom order very unusual hoagies that take a long time to make.

   Harry's regular customers are pushy and demanding. They want hoagies made *exactly* according to their special order, and if their hoagies aren't started as soon as they enter the Hoagie House, they get angry and leave.

   Harry used to have many employees to help him make the hoagies, but unfortunately, all of Harry's helpers went on strike, leaving Harry alone to make the hoagies. Harry wants to make as many hoagies as possible, but can now only make one hoagie at a time.

   Design and analyze a strategy to help Harry quickly decide which hoagies to make. Your strategy should take as input the information about regular customers (when they arrive and how long their hoagie takes to make) and output which hoagies Harry should make. Does your strategy guarantee that Harry's prepares the maximum number of Hoagies? If so, say why. Otherwise, give an example when it does not maximize the number of hoagies made.

2. **Counterfeit Coins.** You are given n coins and a balance scale. To use this scale, you put a number of coins in a pile on the left part of the scale, and a number of coins in a pile on the right. The scale indicates which pile is heavier.

   Most of the coins are identical in every aspect; however, one of the coins is counterfeit and much heavier than the rest. Design an algorithm to identify the counterfeit coin that uses the scale at most $\log_3 n$ times.

3. **Minor League Baseball.** In the past ten years, there has been a renaissance in minor league baseball. Teams have become very successful via clever marketing and creating unique mascots, such as the IronPigs, Isotopes, Sand Gnats, etc.

   A group of n baseball fans $F = \{f_1, \ldots, f_n\}$ are touring n minor league baseball stadiums $S = \{s_1, \ldots, s_n\}$, each in search of a new favorite team over the course of a season (of $m \geq n$ days). Each fan f_i chooses an itinerary where he/she decides to visit one baseball stadium per day. When a fan decides on a team to root for, he/she will stay at that stadium for the rest of the season. However, these fans are pretentious and fiercely independent, preferring not to be fans of the same team. In fact, no two of these fans want to be at the same stadium on the same day.

   Show that it is possible to assign each fan f a unique stadium s_f, such that when f arrives at s_f according to the itinerary for f, all other fans f' have either stopped touring baseball

stadiums themselves, or f' will not visit s_f after f arrives at s_f. Describe an algorithm to find this *matching*.

Hint: The input is somewhat like the input to stable matching, but at least one piece is missing. Find a clever way to construct the missing piece(s), run stable matching, and show that the final result solves the baseball fan problem. You can assume that the itineraries of the fans are such that no two fans will plan on touring the same stadium on the same day. (Of course, if one fan decides to stay at a stadium, there might be a problem.)

4. **The Wedding Planner Problem.** Imagine you are a wedding planner. Among other tasks, you must help your customers with their guest lists. Couples come to you with lists of people they might invite to their wedding. They also come with demands – Alice and Bob need to be invited, but if Bob is invited, do not invite Carol (they have history). When Carol, Dave, and Eve get toegether, they only talk about Justin Bieber (their favorite musician), so don't invite all three. However, any two of them is ok. Call these conditions *constraints*.

   People at weddings are **demanding**. Everything needs to be exactly perfect, and they will blame you if even one constraint is not satisfied. You're not even sure if this is possible, and it when it's not, it would be nice to have a convincing argument for the wedding couple.

   Design an algorithm that takes a list of people, and a list of constraints, and outputs YES if there is an invite list that satisfies every constraint. Otherwise, output NO.