CS41 Lab 1

September 7, 2017

In typical labs this semester, you'll be working on a number of problems. You are encouraged to work in groups of three or four. You will not be handing 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. **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.

2. **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.

3. **Hipster coffee tours.** A group of n Portland hipsters $H = \{h_1, \ldots, h_n\}$ are touring a set of local coffee shops $C = \{c_1, \ldots, c_n\}$ over the course of $m \geq n$ days. Each hipster h_j has an itinerary where he/she decides to visit one coffee shop per day (or maybe take a day off if $m > n$). However, hipsters are fiercely independent and prefer not to share coffee shops with other hipsters. Furthermore, each hipster is looking for a favorite coffee shop to call his or her own. Each hipster h would like to choose a particular day d_h and stay at his/her current coffee shop c_h for the remaining $m - d_h$ days of the tour. Of course, this means that no other hipsters can visit c_h after day d_h, since hipsters don't like sharing coffee shops.

   Show that no matter what the hipsters' itineraries are, it is possible to assign each hipster h a unique coffee shop c_h, such that when h arrives at c_h according to the itinerary for h, all other hipsters h' have either stopped touring coffee shops themselves, or h' will not visit c_h after h arrives at c_h. 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 hipster 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 Beyoncé (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.