CS41 Homework 2

This homework is due at 11:59PM on Wednesday, September 14. Write your solution using LaTeX. Submit this homework in a file named `hw2.tex` using **github**. This is an individual homework. It's ok to discuss approaches at a high level. In fact, we encourage you to discuss general strategies. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else. The only exception to this rule is work you've done with a lab partner *while in lab*. In this case, note (in your **README file**) who you've worked with and what parts were solved during lab.

When you submit homework assignments this semester, please keep the following in mind:

- Don't forget to fill out the README.md file.
- Don't include your name in hw2.tex. (I'd like the graders to *not* know who you are, to minimize grader bias)
- Don't submit a .pdf – just the .tex will do.
- Graders will compile your code from the .tex file using pdflatex. It is your responsiblity to make sure the LaTeX compiles.

The main **learning goals** of this lab are to work with stable matching and the Gale-Shapley algorithm, and get comfortable analyzing it and applying it.

1. **Stable Matching Runtime.** We showed in class that the Gale-Shapely Algorithm for stable matching terminates after at most n^2 iterations of the while loop.

   (a) For two sets A and B of size n, can a particular list of rankings actually result in a quadratic number of iterations? If so, describe what the rankings would look like. If not, argue why no set of rankings would ever result in a quadratic number of iterations. **Note:** the algorithm need not take exactly n^2 iterations, but *asymptotically* n^2 iterations, meaning $0.1n^2$ would be sufficient to show your claim.

   (b) Can a particular set of rankings result in strictly less than a quadratic number of iterations? Can you design an input that requires $O(n)$ iterations? If so, describe the structure of this input. If not, argue why this is not possible.

   (c) Finally, can you design an input that takes fewer than n iterations? Why or why not?

   Aim for clarity and conciseness in your write up of this problem. You should have all the necessary tools to express your solutions. You do not need formal proofs or pseudocode, but you should be able to clearly articulate your ideas in plain English (math notation is also ok as long as it is readable).

2. **Practice with proofs.**

   (a) Sometimes a proof which follows the correct structure can have a subtle problem. Consider the following proof.

   Claim 1. *All dogs are the same color.*

Proof. (by induction)

- base case: Let's say we have just $x = 1$ dog. Then clearly all the dogs we're considering are the same color.
- inductive hypothesis: Assume that if we have $x = k$ dogs in a group, they are all the same color, for some arbitrary constant k.
- inductive step: Let's say we have $x = k + 1$ dogs in a group. We want to show that they are all the same color.
  Let's consider the group if we take out one dog ("Muffles"). Now we have a group of k dogs, so by the inductive hypothesis, they are all the same color.
  Now add Muffles back in, and let's take out a different dog ("Hairy Potter"). Now again we have a group of k dogs, so by the inductive hypothesis, they are all the same color.
  Since the group with Potter but not Muffles is all the same color, and the group with Muffles but not Potter is all the same color, it must be that Muffles and Potter are the same color. Therefore all $k + 1$ dogs in the group are the same color.

Therefore, by induction, all dogs are the same color. □

It seems like something is probably wrong with this argument, since dogs exist in many different colors. Explain what the issue is with this proof.

(b) Prove or disprove the following claim:

Claim 2. *All CS professors wear an outfit consisting of a plaid button-down shirt and khakis when they teach.*

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.

   It may be necessary to break ties; i.e., two hipsters might choose to visit the same coffee shop on the same day. You may assume that the tie can be broken by having hipsters arrive at different times of the day such that if h and h' both want to visit c on the same day, that there is some timestamp on their visits such that it is easy to determine who arrived at c first. Thus, for any given day, at any given coffee shop, there is a well-defined ordering to the planned arrival time of the hipsters.

4. **(extra credit problem)** In class, we discussed a version of the stable matching problem where we want to match n doctors to n hospitals. In this problem, we discuss the homogeneous version. The input is a set of students $A = \{s_1, \ldots, s_{2n}\}$ of size $2n$. Each student ranks the others in order of preference. A homework partner assignment of students into partners $M = \{(i, j)\}$ is a matching; it is unstable if there exists $(i, j), (i', j') \in M$ such that i prefers j' to j and that j' prefers i to i'. It is stable if it is a perfect matching and there are no instabilities.

   (a) Does a stable homework partner assignment always exist? Prove that such an assigment must always exist, or give an example where no stable assignment occurs. (Remember, you must have $2n$ students.)

   (b) Design and analyze an efficient algorithm that either returns a stable matching for homework partners or outputs that no such matching exists.