Algorithms (CS:3330:0001 or 22C:031:001)
Homework 1

Each of the four questions counts for the same number of points. The homework is due in class on Monday, September 8th. If you can't make it to class, drop it in my mailbox in the MacLean Hall mailroom.

1. Consider instances of the stable matching problem with 2 doctors and hospitals.[1] Given a perfect matching M, a particular hospital is either assigned its top choice or not. The same goes for any doctor. Let $n(M)$ denote the number of agents (hospitals and doctors) who are assigned their top choice in M. In this question, we explore the connection between $n(M)$ and whether perfect matching M is stable or not.[2]

   (a) If $n(M) = 1$, is M necessarily unstable? Explain.

   (b) If $n(M) = 3$, is M necessarily stable? Explain.

   (c) If $n(M) = 2$, is M necessarily unstable? Necessarily stable?

2. We showed that for any instance with n doctors and n hospitals, the stable matching algorithm terminates in at most n^2 iterations of the while loop. (Recall that in each iteration, some unmatched hospital proposes to the next doctor on its ordered list of preferences.)

   For any $n > 0$, construct an instance and a corresponding order in which the algorithm makes its proposals, that demonstrate that the algorithm takes $\Omega(n^2)$ iterations in the worst case.

   (This is basically Exercise 3 of Lecture 0 from Jeff Erickson's lecture notes, which is an on-line reference for us. I mention this just as an acknowledgement, and not to suggest you read the lecture notes.)

3. Consider a generalization of the stable matching problem, where some doctors do not rank all hospitals and some hospitals do not rank all doctors, and a doctor can be assigned to a hospital only if each appears in the other's preference list. In this case, there are three additional unstable situations:

   - A hospital prefers an unmatched doctor α to its assigned match, and α has the hospital on her preference list.
   - A doctor prefers an unmatched hospital A to her assigned match, and A has the doctor on its preference list.
   - An unmatched doctor and an unmatched hospital appear in each other's preference lists.

[1] In the textbook, women and men take the place of doctors and hospitals.

[2] This was brought up by someone in class.

Describe and analyze an efficient algorithm that computes a stable matching in this setting. (Here, 'efficient algorithm' means that the running time is at most some polynomial in n, the number of doctors/hospitals.) Note that a stable matching may leave some doctors and hospitals unmatched, even though their preference lists are non-empty. For example, if every doctor lists Harvard as their only acceptable hospital, and every hospital lists Dr. House as their only acceptable intern, then only House and Harvard will be matched.

(This is Exercise 5 from Lecture 0 of Jeff Erickson's lecture notes.)

Hint: I don't know if this strategy will work, but the first thing I would try is a reduction to the regular stable matching problem. That is, see if you can use our subroutine for solving the regular stable matching problem by setting up its input appropriately.

4. You're doing some stress-testing on various models of glass jars to determine the height from which they can be dropped and still not break. The setup for this experiment, on a particular type of jar, is as follows. You have a ladder with n rungs, and you want to find the highest rung from which you can drop a copy of the jar and not have it break. We call this the *highest safe rung.*

   It might be natural to try binary search: drop a jar from the middle rung, see if it breaks, and then recursively try rung $n/4$ or $3n/4$ depending on the outcome. But this has the drawback that you could break a lot of jars in finding the answer.

   If your primary goal were to conserve jars, on the other hand, you could try the following strategy. Start by dropping a jar from the first rung, then the second rung, and so forth, climbing one higher each time until the jar breaks. In this way, you need only one jar – at the moment it breaks, you have the correct answer – but you may have to drop it n times (rather than $\log n$ as in the binary search solution).

   So here is the trade-off: it seems you can perform fewer drops if you're willing to break more jars. Suppose that you are given a budget of 2 jars to break. Describe a strategy for finding the highest safe rung that guarantees at most $f(n)$ jar drops, for some function $f(n)$ that grows slower than linear. (In other words, it should be the case that $\lim_{n\to\infty} f(n)/n = 0$.)

   (This is Exercise 8 (a) from Chapter 2 of the textbook, reproduced here in case not all people are using the same edition.)

   Hint, for those who find asymptotics unfriendly: Think about a concrete n first, say $n = 100$. Then generalize.

   ps: In case you didn't already, assume that all jars of the same model behave identically!

Let's review some policy from the first day handout. "For homework problems, collaboration is allowed, assuming each of you has first spent some time (about 30 minutes)

working on the problem yourself. However, no written transcript (electronic or otherwise) of the collaborative discussion should be taken from the discussion by any participant. Furthermore, discussing ideas is okay but viewing solutions of others is not. It will be assumed that each of you is capable of orally explaining the solution that you turn in, so do not turn in something you don't understand. Students are responsible for understanding this policy; if you have questions, ask for clarification."

Please note that although the correctness of your solution counts for a significant fraction of your score, the quality of your writing and a demonstration that you understood the question and made a serious attempt at solving it also count for a significant fraction.