Homework 1: Proofs and Stable Matchings

Due Date: This assignment is due at 11:59 PM Friday January 15 (Seattle time, i.e. GMT-8).
You will submit the written problems as a PDF to gradescope. Please put each numbered problem on its own page of the pdf (this will make selecting pages easier when you submit). The coding problem will also be submitted to gradescope.

Collaboration: Please read the full collaboration policy. If you work with others (and you should!), you must still write up your solution independently and name all of your collaborators somewhere on your assignment.

FUN-damentals!

The following problems have two goals:

(a) To review content you learned in CSE 373 (or equivalent course) but might have forgotten. We have specifically chosen problems for topics that show up early and frequently in the course.

(b) To (re-)introduce you to proofs. A proof is a rigorous argument – we will need proofs to show that our algorithms actually do the right thing!
You may have written proofs before (say in a course in the math department), or you might not have; we won't be picky about how proofs are formatted, but we will be picky about them communicating the right idea.

1. $\mathcal{O}, \Omega,$ and Θ, oh my! [10 points]

You've analyzed some code, and found that it runs in $f(n) = 5n^2 + 7n + 3$ time.

For each of the following, state whether it is "true" or "false."
You may want to review asymptotic analysis.

(a) $f(n)$ is $\mathcal{O}\left(n^2\right)$

(b) $f(n)$ is $\Omega(n^3)$

(c) $f(n)$ is $\mathcal{O}\left(n^3\right)$

(d) $f(n)$ is $\mathcal{O}\left([\log(n)]^{100}\right)$

(e) $f(n)$ is $\Theta(3n^2)$

2. Recurrence [13 points]

$$T(n) = \begin{cases} 3T\left(n/3\right) + 6n^2 & \text{if n} \geq 9 \\ 2 & \text{otherwise} \end{cases}$$

You may want to review here or here. And you might find the Math resources to have useful formulas.

(a) Use the tree method or unrolling to caclulate the exact closed form of $T(n)$. You may assume n is a power of 3.
If you use unrolling, show at least three applications of $T()$, and a general form of what i levels of unrolling would give.
If you use the tree method, use the step breakdown in the linked materials to organize your work. [11 points]

(b) Check that you got the right leading term by using the Master Theorem. Show the calculation to decide which case of the theorem you are in. [2 points]

3. Proof by Induction [13 points]

Recall Dijkstra's Algorithm from 373:

```
Dijkstra(Graph G, Vertex source)
    initialize distances to infinity
    mark all vertices unprocessed
    set source.dist = 0
    while(there is an unprocessed vertex)
        let u be the closest unprocessed vertex
        for each(edge (u,v) leaving u)
            if(u.dist + weight(u,v) < v.dist)
                v.dist = u.dist + weight(u,v)
                v.predecessor = u
            endIf
        endFor
        mark u as processed
    endWhile
```

Prove the following claim holds for all $i \geq 0$ by induction:

If Dijkstra's is run on a graph with no negative edge weights: at the start of the i^{th} iteration of the while loop, if `u` is the closest unprocessed vertex, `u.dist` stores the shortest path distance from `source` to `u`.

You must use induction for this problem, and you cannot use proof by contradiction for any part of the proof. It's not enough to just trace the code, the point of a proof is to explain why the code is doing the right thing – what does the shortest path from `source` to `u` look like?

Hint: Remember that Dijkstra's is only guaranteed to work if the graph has no negative weight edges, you should use that assumption explicitly somewhere in the proof.
Hint: When doing the inductive step, what does the shortest path to the closest vertex look like? Giving a name to the vertex right before `u` on the shortest path from `source` to `u` made our explanation easier.

4. Contrapositive [7 points]

An "implication" is a statement of the form "if p then q." The "contrapositive" of an implication reverses the order of the implication, and negates both p and q ("if not q, then not p").

For example, the contrapositive of "if it is raining, then I have my umbrella." is "if I do not have my umbrella, then it is not raining."

The "converse" of an implication just switches the order ("if q then p"). For example, the converse of "if it is raining, then I have my umbrella." is "If I have my umbrella, then it is raining."

The contrapositive of an implication and the original implication are equivalent (one is true if and only if the other is). The same is not true of the converse (they might or might not have the same value).

Consider the implication "If G has no negative edge weights, then Dijkstra's algorithm returns the shortest paths of G from the chosen source vertex."

(a) State the contrapositive of the sentence above. [2 points]

(b) State the converse of the sentence above. [2 points]

(c) Give an example of a graph G and a chosen source vertex where the converse is false (and thus different from the original claim, which is true). [3 points]

5. Proof by Contradiction [10 points]

You are one of the n riders being matched to n horses. You are one of k **popular riders**. Every horse has the k popular riders as their k top choices (though they might not agree on the ordering of the k riders).

Use a proof by contradiction to show that if Gale-Shapley is run with riders proposing, you will be matched to one of your top k choices.

Caution: This problem looks superficially similar to worked exercise 1 in the Kleinberg and Tardos textbook – this problem is different!

Stable Matchings

6. Stable Matching I [5 points]

Is it possible for a stable matching to have no agent receive their first choice?
If yes, give an example **and** explain why your example works.
If no, prove why no such matching can exist.

7. Stable Matching II [7 points]

We proved in lecture that every stable matching instance has at least one stable matching, and that some stable matching instances have more than one. Design a family of stable matching instances such that the number of stable matchings is $\Omega(c^n)$ where c is a constant greater than 1.

By a "family" of instances, we mean you should give a description so that for any $n \geq 1$, we could write down the stable matching instance you want us to.

You must justify that your construction works, i.e. why an instance with n agents on each side has as many stable matchings as you claim.

We will give full credit for any $c > 1$ (with a proper argument)
We will give extra credit for any $c \geq 2$ – The best-known c is an irrational number about 2.28, but it's an open problem whether that c is actually the biggest possible!

8. Stable Matching Modeling [10 points]

You have a set of n job applicants and m jobs available. Both the job applicants and the companies offering the jobs *have standards*. Each job applicant can declare some (or none, or all) of the jobs as "unacceptable" – that is, they would rather have no job (i.e., not be matched at all) than be matched to an unacceptable job. Similarly, every job can declare some (or none, or all) of the applicants "unacceptable."
Every applicant and job would prefer to be matched to any acceptable option than to be left unmatched.

In this context, call an assignment "stable" if:

- No applicant is matched to a job they declare unacceptable.
- No job is matched to an applicant they declare unacceptable.

- There is no non-matched job-applicant pair that both declare the other acceptable and who both prefer each other to their current state.

Note that we do not require a perfect matching in this context to have a matching be stable – now that we have unacceptable pairings and differing numbers of applicants and jobs, we may leave some agents unmatched.

(a) Given n job applicants and m jobs, along with all of their preference lists and all their decisions as to whether other agents are unacceptable, describe how to use a **standard Stable Matching** algorithm to find a stable assignment for this problem.

(b) Prove that your output produces a stable assignment.

(c) What is the running time of your algorithm? Assume that on an input with n riders and n horses, your standard SM algorithm runs in time $\Theta\left(n^2\right)$. Justify your answer.

Programming

9. Write the code! [10 points]

The goal of this problem is to practice the kind of programming problems you'll see in this course and perhaps an interview. There will be some major differences between this course and 373:

- You'll have less starter code – we aren't training you to integrate your code into existing code bases, like we were in 373.
- You'll be allowed to use data structures libraries instead of always writing your own.

An autograder is provided on Gradescope as a sanity check. However, it does not guarantee complete correctness and its score is subject to adjustment.

In this question, you are asked to complete a method that takes in a stable matching instance and outputs an extreme matching most optimal to one particular side.

Input:

- A 2D array denoting the preferences of the horse. For example, arr[i][j] lists the j-th preferred rider of the i-th horse.
- A 2D array denoting the preferences of the rider.
- Whether if the matching provided should be horse-optimal or rider-optimal.

Output: An array denoting the matching in terms of the horse, namely, arr[i] contains the matched rider of the i-th horse.

(a) Briefly explain what algorithm you will be implementing and how you plan to meet the "extreme matchings" requirement in your code. 2-3 sentences would suffice.

(b) Download the starter code StableMatching.java and complete the method marked "TODO". Then, submit StableMatching.java to the separate assignment on Gradescope. Do not submit it with your written portion.

A sample input is provided in the starter file, which you can take advantage by running the Java class:

```
javac StableMatching.java && java StableMatching
```

You can also add in your own test cases in the main method.

If you don't have Java on your computer, click here to download it. Ask on Ed and Office Hours if you need help with setting up your computer!