CS 5854 : Networks, Crowds, and Markets
Homework 1

Instructor: Rafael Pass Due Date: September 13, 2017

August 30, 2017

Homework Policies and Guidelines:

You are free to collaborate with other students on the homework, but *you must turn in your own individually written solution and you must specify the names of your collaborators.* Additionally, you may make use of published material, provided that you acknowledge all sources used. Note that it is considered a violation of academic integrity to submit a problem solution that you are unable to explain orally to a member of the course staff.

Submit hardcopy or to the TA by the beginning of class on the due date, or submit a .pdf file in CMS. If you wish, you may submit a .zip file containing code files for any coding questions and a single .pdf file with solutions for the remaining questions. Typed problem sets are **strongly** preferred. Each student may use four "late days", each of which grants a 24-hour extension to an assignment's due date, as desired throughout the semester; late work beyond this limit will still be accepted and graded until solutions or grades have been released for that assignment, but (unless discussed in advance with the TA and/or instructor) will have a negative impact on final grades.

For non-coding questions, responses will be graded not only on the correct answer, but also on whether an answer is properly justified (this also means that an incorrect answer with good reasoning or partial justification will be given considerable partial credit). Unless explicitly stated otherwise, please provide justification for your response to every question, and write out the steps of any non-trivial mathematical derivation. (For easy questions, a sentence or two will generally suffice!)

For coding questions, please turn in complete code (it should be executable by the grader) for each part of the question that asks for an implementation, and execution traces or test results for each part of the question that asks for them. Unless it is explicitly specified, you should submit something for each part of every coding question. We will accept code in any (human-readable) language, and we will not grade code based on style, but *we may mark down code if we are unable to understand what it is doing.* In addition, you may use libraries to implement data structures such as graphs, but, unless otherwise specified, you may not use pre-existing implementations of any algorithms without express permission from the TA or instructor.

Part 1: Game Theory

1. For each of the following three two-player games, find (i) all strictly dominant strategies, (ii) the action profiles which survive iterative removal of strictly dominated strategies, and (iii) all pure-strategy Nash equilibria.

(a)

	$(*, L)$	$(*, R)$
$(U, *)$	(4, 3)	(3, 4)
$(D, *)$	(3, 3)	(0, 0)

(b)

	$(*, L)$	$(*, R)$
$(U, *)$	(3, 3)	(3, 2)
$(D, *)$	(4, 3)	(0, 4)

(c)

	$(*, L)$	$(*, R)$
$(U, *)$	(4, 3)	(2, 3)
$(D, *)$	(3, 2)	(1, 1)

2. Consider the two-player game given by the following payoff matrix:

	$(*, L)$	$(*, M)$	$(*, R)$
$(t, *)$	(0, 3)	(6, 2)	(1, 1)
$(m, *)$	(2, 3)	(0, 1)	(7, 3)
$(b, *)$	(5, 2)	(4, 2)	(3, 1)

   (a) Does either player have a strictly dominant strategy? If so, which player and what strategy? If not, what is the smallest number of entries in the payoff matrix which would need to be changed so that some player did have a strictly dominant strategy?

   (b) What are player 1's and player 2's best-response sets given the action profile (m, L)?

   (c) Find all pure-strategy Nash equilibria for this game. Describe how best-response dynamics converges to the pure-strategy Nash equilibria.

3. (a) Prove the following: If player 1 in a two-person game has a dominant strategy s_1, then there is a pure-strategy Nash equilibrium in which player 1 plays s_1 and player 2 plays a best response to s_1.

   (b) Is the equilibrium from part (a) necessarily a *unique* pure-strategy Nash equilibrium?

   (c) In particular, can there also exist a pure-strategy Nash equilibrium where player 1 does not play s_1?

   (d) If s_1 is instead a *strictly dominant* strategy for player 1, how do the answers to (a)-(c) change?

4. Formulate a normal-form game (as a payoff matrix) that has a unique pure-strategy Nash equilibrium, but for which best-response dynamics does not converge. Briefly justify your answer. (*Hint:* Rock-paper-scissors has no equilibrium, and thus is such that BRD will not converge. Try combining it with a game that has an equilibrium.)

Part 2: Graph Theory

Note: Unless stated otherwise, please assume for any problem involving graphs that we refer to *undirected* and *unweighted* graphs.

5. Given a graph, we call a node x in this graph *pivotal* for some pair of notes y and z if x lies on every shortest path between y and z.

   (a) Give an example of a graph in which every node is pivotal for at least one pair of nodes. Explain your answer.

   (b) For for any integer $c \geq 1$, construct a graph where every node is pivotal for at least c different pairs of nodes. Explain your answer.

(c) Give an example of a graph having at least four nodes in which there is a single node x which is pivotal for every pair of nodes not including x. Explain your answer.

6. Given some connected graph, let the *diameter* of a graph be the maximum distance (i.e. shortest path length) between any two nodes. Let the *average distance* be the expected shortest path length between a randomly selected pair of nodes.

   (a) Give an example of a (connected) graph where the diameter is more than three times as large as the average distance. Justify your answer.

   (b) Describe how you could extend your construction to produce (connected) graphs in which the diameter exceeds the average distance by an arbitrarily large factor. (That is, for every number c, can you produce a graph in which the diameter is more than c times as large as the average distance?)

7. Consider a graph on n nodes.

   (a) What is the fewest number of edges that can exist in this graph so that it is connected?

   (b) What is the fewest number of edges that can exist in this graph so that any two nodes have a shortest path length of 1? Prove that this is the minimum.

   (c) Repeat part (b) for a shortest path length of at most 2.

Part 3: Coding: Shortest Paths

8. Submit the following:

   (a) Implement (and turn in) a procedure that produces an undirected graph with a large number of nodes (say, 1,000), and connects each pair of nodes by an edge with some probability p (specified as an argument).

   (b) Implement a general shortest-path algorithm for graphs, as described in lecture, that works on your data set. (Turn in your algorithm as part of your homework submission.) Make sure to handle the case where the graph is disconnected (i.e. no shortest path exists), either by outputting "infinity" or a suitably large number.

   (c) Run your shortest-path algorithm on a graph with $p = 0.1$ many (10,000 or more) times using random pairs of nodes and use it to calculate an estimate of the average shortest path between two nodes in this data set. (Turn in code detailing how you implemented this and a small segment—say, 100 searches—of an execution trace of your code, preferably printing (node A, node B, path length) for each pair and the final average at the end.)

   (d) Run the shortest-path algorithm on data sets constructed with many values of p (for instance, 0.01 to 0.04 using .01 increments, and then 0.05 to 0.5 using .05 increments). Turn in your numerical data, and plot the average shortest path as a function of p.

   (e) Intuitively explain the behavior of the data you found; specifically, as p increases (in particular, look at the larger values, e.g. 0.3 and above), what function does the average shortest path length seem to asymptotically approach and why?

9. Now run your code on the Facebook social network data available at: http://snap.stanford.edu/data/egonets-Facebook.html

   (In particular, please refer to the file "facebook_combined.txt.gz"; the data is formatted as a list of undirected edges between 4,039 nodes, numbered 0 through 4038. You will need to parse this data as part of your code; knowing how to do this will be useful for subsequent assignments!)

   (a) Repeat the same analysis as in part c) (i.e. run your algorithm on 10,000 random pairs of nodes and determine the average shortest path length). Turn in your parsing code and a small segment of your execution trace, along with the final average you computed.
   (b) What is p for the Facebook data? That is, given two random nodes, what is the probability they are connected by an edge? (You may compute this either precisely or empirically, but you will likely find the former to be easier.)
   (c) Given your responses to questions (8d), (9a), and (9b), is the average shortest path length of the Facebook data greater than, equal to, or less than you would expect it to be if it were a random-edge graph with the same value of p? (To answer this, you may wish to run your code from question (8c) using the p you determined in part (9b).) Explain why you think this is the case.