CS 5854 : Networks, Crowds, and Markets
Homework 1

Instructor: Rafael Pass TAs: Cody Freitag, Ben Chan

Due: September 22, 2020, 11:59 pm Eastern Time

Homework Policies and Guidelines:

Submission: Your homework solutions must be **typed** and submitted as a single .pdf file on Canvas. You must additionally submit a single .zip file containing all relevant files specified in the assignment for all coding problems. Template .tex and .py files will be provided containing an outline for your submission.

Late Days: Each student may use four "late days" in total as desired throughout the semester, each of which grants a 24-hour extension to an assignment's due date. Late work beyond this limit will still be accepted and graded until 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.

Collaborations: You may work in groups of up to **3** students. Every member of the group must list all other collaborators at the top of their assignment. *(Note: the maximum size of a connected component of groups must be 3. If A,B, and C work together, it must not be the case that A and D work together.)*

Your submitted **answers, explanations, and discussion** for all written questions in the pdf **must be your own, individual, unique solutions**. You will receive **ZERO** points for written explanations which are copied verbatim or copied with minor changes (up to the discretion of the TAs)—either from another group member or from a *cited* online source. You *may* share and even submit the same code, examples, figures, and graphs with your collaborators, but your explanations must be your own. Additionally, you may make use of published material (papers, github, wikipedia, etc.), provided that you acknowledge and specifically cite all sources used. Still, this does not give you permission to copy code/ explanations from an online source. 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.

How to receive credit: You must **justify all answers** to receive credit unless specified otherwise. We will do our best to make clear the level of justification we expect for each problem. For coding questions, please turn in complete, executable code for each part of the question that asks for an implementation, and include a .txt file containing any required outputs if not already included in the main .pdf file. *Using Python is required.* We will not grade code based on style, but we may mark down code if we are unable to understand what it is doing. You may use standard 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 TAs. (If a problem asks you to implement X and you use a package that implements X for you, you will not get credit.)

Grading: There will be four homeworks throughout the semester. Each homework (and each problem within each homework) will receive an associated weight specified by a number of points. Your raw score at the end of the semester will be the sum of points earned divided by the total number of available points. If your raw score is greater than 94%, you are guaranteed to get at least an A, 90% for A-, 87% for B+, 84% for B, and so on. We reserve the right to lower the cutoffs (but we will not raise them).

There will additionally be optional bonus problems on some assignments. These will not be factored into your raw point totals, but will be factored in after computing your raw score to determine your final grade (for getting an A+, or possibly bumping up 1 or 2 letter grades).

Part 0: Logistics

Slack: Join the slack channel for the course, via the following link: SLACK

Policies: Read the full homework policies and guidelines above. If there are any questions, please ask them on slack before this first homework is due.

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. Give a brief justification for each part.

(a)

	$(*, L)$	$(*, R)$
$(U, *)$	(5, 4)	(4, 5)
$(D, *)$	(4, 4)	(0, 0)

(b)

	$(*, L)$	$(*, R)$
$(U, *)$	(2, 2)	(2, 1)
$(D, *)$	(3, 2)	(0, 3)

(c)

	$(*, L)$	$(*, R)$
$(U, *)$	(6, 5)	(4, 5)
$(D, *)$	(5, 4)	(2, 2)

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

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

   (a) Does either player have a strictly dominant strategy? If so, which player, what strategy, and why? 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? Justify why this is the minimum, i.e. there is no smaller value that works.

   (b) What are player 1's and player 2's best-response sets given the action profile (m, L)? Explain in words why these are the best responses.

   (c) Find all pure-strategy Nash equilibria for this game. (Argue why all that you wrote are PNEs and why there are no others.) Describe how best-response dynamics might converge to each 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. (Your proof should be near the level of formality used in the book. Part of the point of this problem is to get familiar with formally arguing about the definitions provided.)

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

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

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

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 always converge (i.e. there are possible starting states for which BRD will not converge). Justify your answer. (*Hint:* Try modifying a game where BRD does not converge to give it a unique PNE.)

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 nodes y and z if x (not equal to y or z) 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 any integer $c \geq 1$, construct a graph where every node is pivotal for at least c different pairs of nodes. That is, if I give you any value for $c \geq 1$, your explanation should tell me how to construct such a graph for that c. 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 distinct nodes.

   (a) Let G be a graph with average distance A. What is the smallest diameter possible for such a graph? Provide a graph G that attains this minimum and prove that any smaller is impossible.

   (b) Give a graph G with average distance A and diameter at least $3 \cdot A$.

   (c) Repeat (b) for a diameter of at least $100 \cdot A$. (You don't need to draw the graph, just describe it and briefly justify why the diameter is at least 100 times larger than than the average distance.) Describe how you could extend this to an arbitrarily large factor $C \cdot A$.

   (d) Discuss what the diameter and average distance of a social network (given as a graph) might represent. What might it mean if the diameter is very similar to the average distance? What might it mean if the diameter is much greater?

7. Consider a graph G on n nodes.

(a) What is the fewest number of edges such that G is connected? Give an example with that many edges, and argue why any fewer edges must result in a graph G which is disconnected.

(b) What is the fewest number of edges such that any two nodes in G have a shortest path length of 1? Again, prove that this is the minimum by arguing that no fewer is possible and that the number you give is attainable.

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

Part 3: Coding: Shortest Paths

8. Submit the following:

   (a) In graph.py, implement (and turn in) a function create_graph(n,p) that produces an undirected graph with n nodes where each pair of nodes is connected by an edge with probability p.

   (b) Implement a general shortest-path algorithm for graphs, as described in lecture, that works on your graph. In graph.py, include a function shortest_path(G,i,j) that outputs the length of the shortest path from node i to j in your graph G. Make sure to handle the case where the graph is disconnected (i.e. no shortest path exists) by outputting "infinity".

   (c) Construct a graph for $n = 1000$ and $p = 0.1$. Estimate the average shortest path between a random pair of two (connected) nodes in the graph. For accuracy, repeat for 1000 random pairs of nodes in your graph. Output an execution trace in avg_shortest_path.txt containing all path lengths written as (i, j, length).

   (d) For $n = 1000$, run the shortest-path algorithm on data sets for 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 as varying_p.txt, and plot the average shortest path as a function of p and submit as an image file varying_p.(image extension) or include in your main .pdf file.

   Note: For $p = 0.01$ there is actually a small but reasonable chance (around 4%) to produce a disconnected graph. If this occurs, resample and produce a connected graph for the purposes of gathering data.

   (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 $f(p)$ does the average shortest path length asymptotically approach? Justify why it behaves this way.

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 8(c) (i.e. run your algorithm on 1000 random pairs of nodes and determine the average shortest path length). Include your code in graph.py and include an execution trace in fb_shortest_path.txt

(b) For the Facebook data, estimate the probability p that two random nodes are connected by an edge. Explain how you computed p.

(c) 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 graph with the same number of nodes and value of p? (To answer this, you may wish to run your code from question (8c) using the p you determined in part (9b) and 4039 nodes.) Explain why you think this is the case.