CS 5854 : Networks, Crowds, and Markets
Homework 1

Instructor: Rafael Pass TAs: Cody Freitag, Drishti Wali

Assigned: September 5, 2019 Due: September 24, 2019, 11:59 pm

KUSHAL SINGH

Collaborators: I worked with Chris Shei.
Outside resources: I uses the following outside resources: https://codereview.stackexchange.com/questions/193410/breadth-first-search-implementation-in-python-3-to-find-path-between-two-given-n.
Late days: I have used 0 many late days on this assignment.

Part 0: Slack

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

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)

Solution:

(a) i) For Player 1, moving Up (U) is strictly dominant. If Player 2 chooses Right (R), then Player 1 would want to choose U, since its utility of 4 is greater than the utility of moving Down (D), which is equal to 0. Similarly, if Player 2 chooses Left (L), then Player 1 would want to choose U, since its utility of 5 is greater than the utility of moving Down (D), which is equal to 4.

For Player 2, there are no strictly dominant strategies. This is because if Player 1 chooses U, then Player 2 would want to choose R, since $5 > 4$. However, if Player 1 chooses D, then Player 2 would want to choose L, since $4 > 0$.

ii) (U, R) with utility (4, 5) survives iterative removal of strictly dominated strategies. Since D is strictly dominated by U for Player 1, we can cross out the D row, which eliminates action profiles (D, L) and (D, R). This causes L to be strict dominated by R, since the utility of 5 is greater than the utility of 4. This leaves us with (U, R) as the action profile that survives.

iii) (U, R) with utility (4, 5) is the Nash Equilibrium in this case. This can be derived from the iterative removal of strictly dominated strategies in part ii).

(b) i) In this scenario, there are no strictly dominant strategies for either player. This is because if Player 1 chooses U, then Player 2 would choose L, whereas if Player 1 chooses D, then Player 2 would choose R. Likewise, if Player 2 chooses L, then Player 1 would choose D, but if Player 2 chooses R, then Player 1 would choose U.

ii) Following from part i) above, all action profiles survive iterative removal of strictly dominated strategies, since there are no strictly dominated strategies for either player.

iii) There are no pure-strategy Nash equilibria for this problem, since the best-response actions move in a counterclockwise circle without converging. For example, if Player 1 chooses D, then Player 2 will choose R. This will cause Player 1 to choose U, which will then cause Player 2 to choose L. This will, in turn, cause Player 1 to choose D, leading to a cycle.

(c) i) For Player 1, moving Up (U) is strictly dominant. If Player 2 chooses Right (R), then Player 1 would want to choose U, since its utility of 4 is greater than the utility of moving Down (D), which is equal to 2. Similarly, if Player 2 chooses Left (L), then Player 1 would want to choose U, since its utility of 6 is greater than the utility of moving Down (D), which is equal to 5.

For Player 2, there are no **strictly** dominant strategies. However, playing L is a dominant strategy.

ii) Following from part i) above, we know that D is strictly dominated by U, so we can eliminate (D, L) and (D, R), leaving us with (U, L) and (U, R) as the action profiles that survive iterative removal of strictly dominated strategies.

iii) (U, L) with utility (6, 5) and (U, R) with utility (4, 5) are the pure Nash Equilibrium for this payoff matrix. We can come to this conclusion by looking at the best-response sets for both players. If we look at the (*, L) and (*, R) columns, we circle 5 and 5, respectively. If we look at the (U, *) and (D, *) rows, we circle 6 and 4, respectively. Since (U, L) and (U, R) contain the best-response choices for Players 1 and 2, these are the two equilibria.

□

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)?

(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.

Solution:

(a) Neither player has a strictly dominant strategy. For Player 1, the smallest number of entries in the payoff matrix which would need to be changed in order for Player 1 to have a strictly dominant strategy would be 2. This is because the highest utility actions (5, 6, 4) are all in different rows. In other words, t would be the best action only if Player 2 were to choose M, m would be the best action only if Player 2 were to choose R, and b would be the best action only if Player 2 were to choose L. In order to make any one action (t, m, or b) preferable, regardless of the other player's choice (i.e. strictly dominant) we would have to change at least 2 entries in the payoff matrix. For example, if we were to make t as a strictly dominant strategy, we would need to change (m, M)'s utility to be (¿ 6, 0). We would also need to change (b, M)'s utility to be (¿ 4, 1). A similar line of reasoning can be applied if, instead, we were to make m or b a strictly dominant strategy.

Similarly, for Player 2, the smallest number of entries in the payoff matrix which would need to be changed in order for Player 1 to have a **strictly** dominant strategy would be 2. This is because there are a couple ties in the maximum utility values. In other words, if Player 1 chose m, then Player 2 could either choose L or R, since either choice would yield a utility of 2. Similarly, if Player 1 chose b, then Player 2 could either choose L or M, since either choice would yield a utility of 1.

(b) Given the action profile (m, L), player 1's best-response set is to play b, since moving to b would yield a utility of 4, which is greater than staying at m, which would yield a utility of 1.

Player 2's best-response set is to stay, since playing M would yield a lower utility of 0, while playing R would yield an equal utility of 2.

(c) The pure-strategy Nash equilibria for this game are (b, L) and (m, R). We can come to this conclusion by looking at the best-response sets for both players. For Player 2, if we look at the $(*, L)$ column, we consider both $(t, *)$ and $(m, *)$ as best-responses since both yield the highest utilities of 2. If we look at the $(*, M)$ column, we consider both $(t, *)$ and $(b, *)$ as best-responses since both yield the highest utilities of 1. If we look at the $(*, R)$ column, we consider $(m, *)$ as a best-response since this yields the highest utility of 2.

For Player 1, if we look at the $(t, *)$ column, we consider $(*, M)$ as a best-response since this yields the highest utility of 5. If we look at the $(m, *)$ column, we consider $(*, R)$ as a best-response since this yields the highest utility of 6. If we look at the $(b, *)$ column, we consider $(*, L)$ as a best-response since this yields the highest utility of 4.

Looking at the intersection between the best-response sets for both players, we see that this occurs for (b, L) and (m, R).

$\square$

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? 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.

Solution:

(a) Assume that player 2's best response strategy is s_{BR}. Then, by definition, a PNE is one in which both players play the best-response to one another's action. In this case, player 1's best response is s_1, since this is a dominant strategy, which means that any other strategy for player 1 must be less than or equal to s_1 in terms of utility. Player 2's best response is s_{BR}, so the best response set is $\{s_1, s_{BR}\}$. This best response set is, therefore, the pure-strategy Nash equilibrium.

(b) The equilibrium from part (a) is **NOT** necessarily a unique pure-strategy Nash equilibrium because player 1 could have multiply dominant strategies that yield a tie for the maximum possible utility attained. Let's say that one such action is $s1'$. Furthermore, player 2's best response set could also have multiple actions that yield a tie for the maximum possible utility attained. Let's say that one such action is $s2'$. If player 1 plays $s1'$ and player 2 plays $s2'$, then this would also constitute a PNE, thus implying that there could be multiple pure-strategy Nash equilibria.

(c) From part (b) above, we can see that there can exist a pure-strategy Nash equilibrium where player 1 does not play $s1$. Namely, since the condition is not **strict** dominance, there can exist multiple dominant strategies that yield a tie for the maximum possible utility attained for player 1. Let's say that one such action is $s1'$. Furthermore, player 2's best response set could also have multiple actions that yield a tie for the maximum possible utility attained. Let's say that one such action is $s2'$. If player 1 plays $s1'$ and player 2 plays $s2'$, then this would also constitute a PNE, thus showing that there exists a pure-strategy Nash equilibrium where player 1 does not play $s1$. be multiple pure-strategy Nash equilibria.

(d) If s_1 is instead a *strictly dominant* strategy for player 1, then parts (a) and (b) stay the same, but the answer to part (c) changes to no.
We can show that part (c) changes to no with a proof by contradiction. Suppose that there exists a pure-strategy Nash equilibrium where Player 1 does not play s_1. Let's call this other action s_other. Then, by definition, s_other must be in the best-response set of player 1. This means that Player 1's best-response set contains s_1, s_other. However, the problem statement states that player 1 has a strictly dominant strategy s_1, which, by definition, means that there cannot exist another strategy which is better than or equal to s_1, thus illustrating a contradiction.

$\square$

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:* Rock-paper- scissors has no equilibrium, and thus BRD will not converge. Can you combine this with a game that does have an equilibrium?)

Solution:

col1	col2	col3	col4
row1	(2, 3)	(0, 0)	(0, 0)
row2	(0, 0)	(-1, 1)	(1, -1)
row3	(0, 0)	(1, -1)	(-1, 1)

In this payoff matrix, if our starting state is (row2, col2) with a utility pairing of (0, 0), and if player 1 (i.e. the row player) starts off, then player 1 would want to move to (row2, col4) since this would yield the highest utility in row2 ($1 > -1 > 0$). Then, player 2 would want to move to (row3, col4) with a utility pairing of (-1, 1) since this would yield the highest utility in col4 ($1 > -1 > 0$). Subsequently, player 1 would want to move to (row3, col3) with a utility pairing of (1, -1) since this would yield the highest utility in row3 ($1 > -1 > 0$). Then, player 2 would want to move to (row2, col3) with a utility pairing of (-1, 1) since this would yield the highest utility in col3 ($1 > -1 > 0$). Then, player 1 would want to move to (row2, col4) since this would yield the highest utility in row2 ($1 > -1 > 0$), yielding a cycle, thus demonstrating that this is a possible starting state for which BRD does not converge.

If we start from any other entry in the matrix, apart from (row1, col2) with utility pairing of (2, 3), then we will enter this cycle, and not reach a Nash equilibrium.

However, if our starting state is at (row1, col2) with a utility pairing of (2, 3), and if player 1 starts off, then player 1 would want to stay at (row1, col2), since this would yield the highest utility in col2 ($2 > 0 > 0$). Then, player 2 would want also want to stay at (row1, col2), since this would also yield the highest utility in row1 ($3 > 0 > 0$). This is a Nash equilibrium.

Putting these results together, we observe that the only Nash equilibrium is at (2, 3), which occurs if the starting state is (2, 3). Any other starting state would result in BRD not converging and, thus, there are no other Nash equilibria.

Since there are no other Nash equilibria, (2, 3) is a pure strategy Nash equilibrium.

Through this example, we have shown that there exists a payoff matrix that has a unique pure-strategy Nash equilibrium, but for which BRD does not always converge.

$\square$

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.

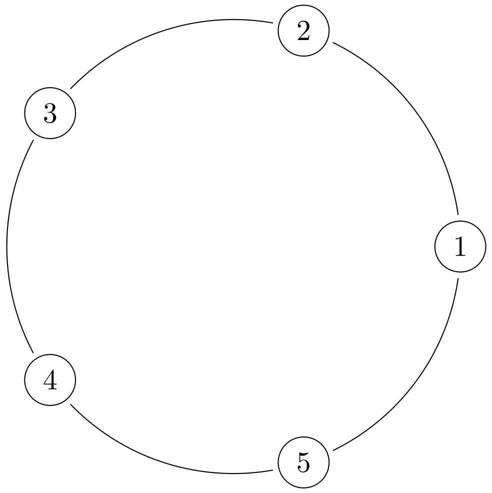
   (a) Give an example of a graph in which every node is pivotal for at least one pair of nodes. Explain your answer.

   (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 $c \geq 1$, you should be able to give me such a graph. 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.

Solution:

(a) In the graph below, we can see that...
1 is pivotal for the pair (2,5).
2 is pivotal for the pair (1,3).
3 is pivotal for the pair (4,2).
4 is pivotal for the pair (3,5).
5 is pivotal for the pair (1,4).



(b) In order to make every node pivotal for at least c different pairs of nodes, we can construct a graph with the number of nodes equal to $3 + 2c$, arranged in a circle where each node has one undirected edge from its predecessor and one undirected to its successor.

Let's imagine we have some arbitrary node A, and another node $B = A + c + 1$. The number of nodes on the top route $= c$ nodes, and the number of nodes on the bottom route $= c + 1$. Therefore, node $A + 1$ is pivotal for all pairs of nodes from $(A, A + 2)$ to

(A, B), which constitutes c different pairs of nodes. Since this graph is a circular evenly weighted graph, by symmetry, this line of reasoning can be applied to every other node in the graph as well.

See FIG 1 on the next page.

(c) See graph with nodes A, B, C, D, E on the next page. Since E lies on the shortest path between any two nodes different than E, we can say that E is pivotal for every pair of nodes.

$\square$

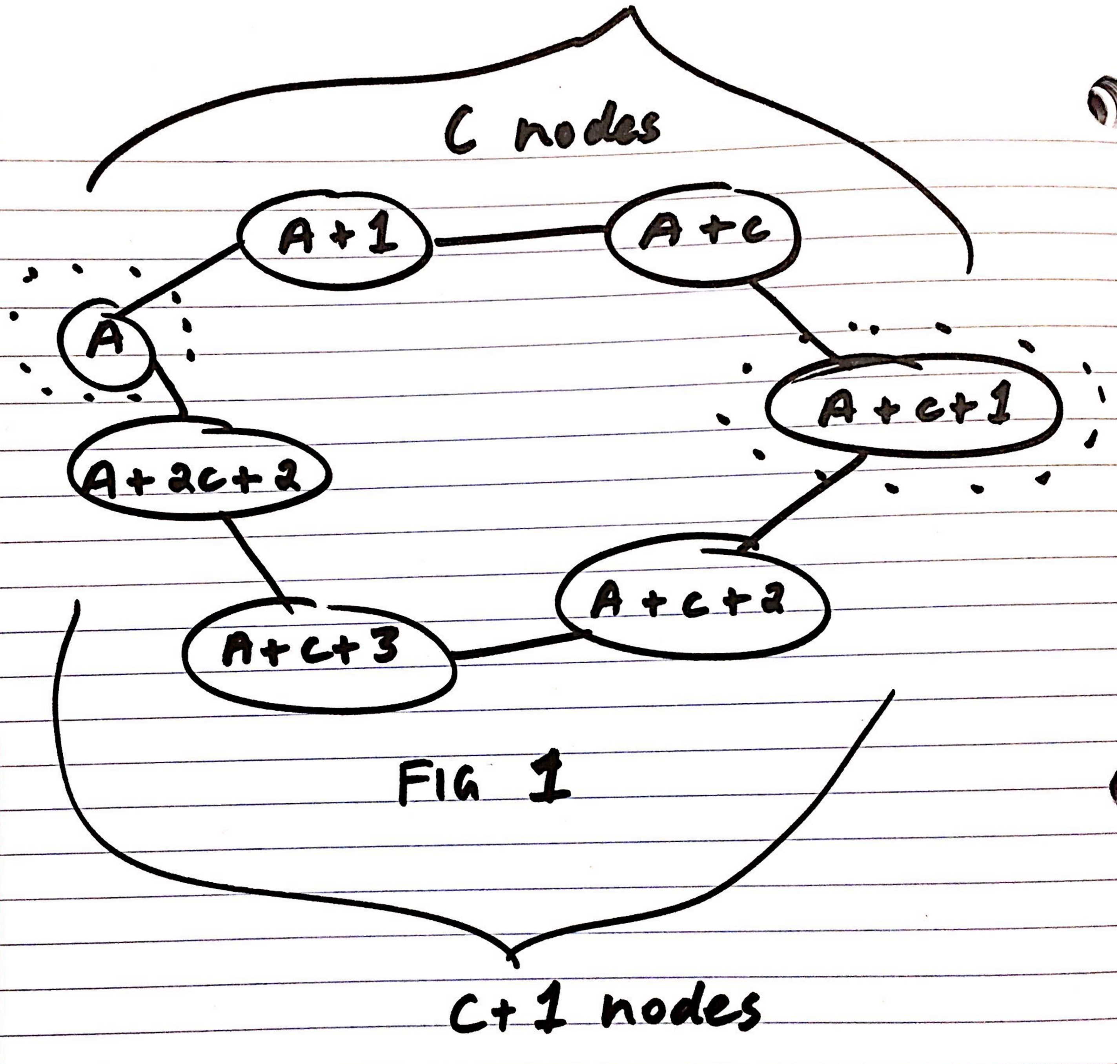
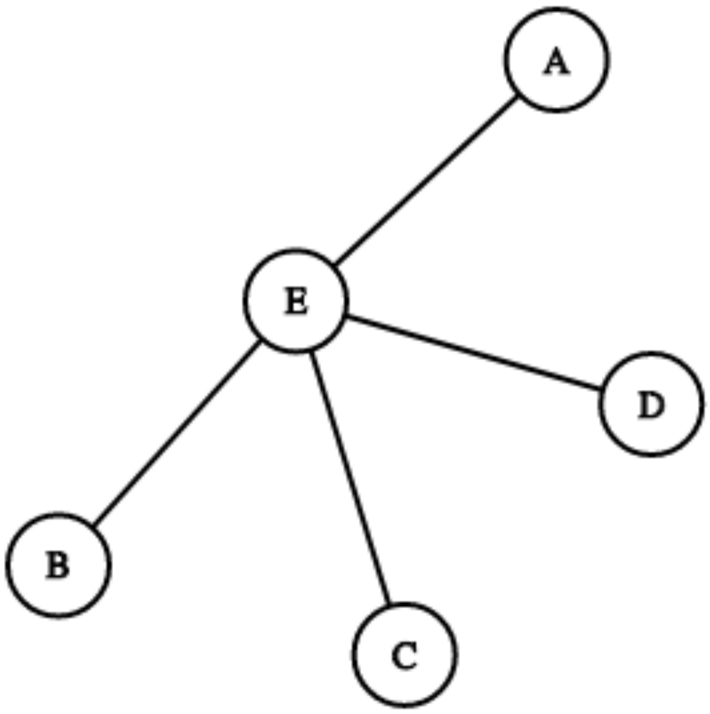


Fig 1


A
E
D
B
C

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 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?

Solution:

(a) The smallest diameter possible for such a graph would be equal to the average distance A.

Let's assume for the sake of contradiction that there exists a diameter $D_{other} < A$. We can define the total shortest distance D_{min} as the sum of all shortest paths between two nodes. We can also define the total shortest distance as the average distance between two nodes * total number of paths in the graph, P_{total}. In other words, $D_{min} = A * P_{total}$.

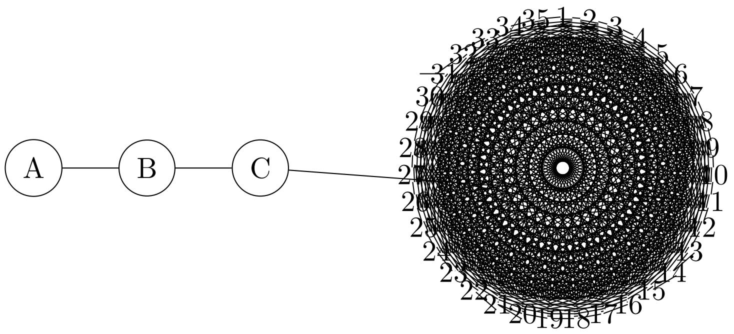
Since D_{other} represents the longest shortest path in the graph, if we multiply D_{other} by P_{total}, then $D_{other} * P_{total}$ should be >= to D_{min}. The case where D_{min} equals $D_{other} * P_{total}$ is the case where all nodes are maximally separated, which is the worst possible case for shortest paths between nodes in a graph.

Putting the equations above together, we see that $D_{other} * P_{total} \geq D_{min} = A * P_{total}$. Simplifying both sides, we see that $D_{other} \geq A$, but this contradicts the original assumption that $D_{other} < A$, thereby implying that the smallest diameter possible for such a graph is indeed A.

(b) The graph is pictured below the justification for this problem. As we can see, the big cluster toward the right hand side of the graph is a maximally connected graph (i.e. a graph where every node is connected to every other node). The diameter is 4, because in order to get from node A to any other node in the maximally connected subgraph we must first traverse the following 3 edges: $A - B, B - C, C - 11$. Then, the extra edge from 11 to whichever node we wish to go to constitutes an additional edge, resulting in 3 + 1 = 4 edges. To calculate the average distance, A, we simply count up all the shortest edge paths and divide that by the number of nodes choose 2, to represent the odds that you choose an edge between any two random nodes. In our case, we have 38 total nodes (35 within the cluster, 3 outside the cluster). Below is a breakdown of the sum of all shortest paths:

The total number of paths with a cost of 4 constitute all shortest paths between A and

every node in the cluster, excluding 11. This is equal to 4(34) = 136.
The total number of paths with a cost of 3 constitute all shortest paths between B and every node in the cluster, excluding 11. It also includes the path from A-B-C-11. This is equal to 3(34) + 1 = 103.
The total number of paths with a cost of 2 constitute all shortest paths between C and every node in the cluster, excluding 11. It also includes the path from A-B-C-11. This is equal to 2(34) + 1 = 69.
The total number of paths within the cluster is equal to $\frac{n*(n-1)}{2} = \frac{35*34}{2} = 595$.
We must also include the single length edges $(A-B, B-C, C-11)$, which sums to 3.
Therefore, the sum of shortest paths is equal to 136 + 103 + 69 + 595 + 3 = 906.
Then, $A = \frac{906}{\binom{38}{2}} = 1.28876244666$, which is roughly equal to 1.3.
From this, we can see that $D \geq 3 * A$, since $4 >= 3 * 1.3 = 3.9$.
The key insight here is that by continuously adding more nodes to the cluster, you could theoretically asymptotically decrease the average distance (getting it infinitely close to 1).



(c) To achieve a diameter of at least 100 * A, we can simply leverage the insights gained from the previous part. Namely, we could keep adding nodes to the maximally connected cluster, thus driving the average distance asymptotically toward 1. Then, we could attach 100 nodes in a straight line (similar to nodes A, B, and C in the graph above) to one of the nodes in the mass (if we are using the graph from above as an example, then it would be node 11). This would give us a diameter of 101 (100 to get from the leftmost edge in the straight line to node 11), and then one additional edge to get from node 11 to any other node in the maximally connected subgraph. Since A is asymptotically close to 1, we would have a network that has a diameter 101 >100 * ~ 1.
A similar logic (i.e. continuing to add nodes in a straight line) can be applied to an arbitrarily large factor $C * A$.

(d) In a social network, diameter represents how far apart the two most distant people in a network are. Average distance shows, on average, how tightly knit the network is. If the diameter is roughly equal to the average distance, then we have a relatively evenly spaced graph that is fairly close knit. However, if the diameter is much greater than the average distance, then we would have a very tight knit network with several outliers that could very easily be maximally disconnected from the main cluster, thus causing the diameter to increase a lot, relative to the average distance.

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.

Solution:

(a) Given a graph G on n nodes, the fewest number of edges such that G is connected would be $n-1$. We can show this with a proof by induction. Consider the base case when $n = 2$. In this situation, the fewest number of edges such that G is connected would be 2-1=1, otherwise the graph would be disconnected. If we were to add one more node to the graph, the minimum number of edges would increase by 1, since the previous graph was already minimally connected, and simply adding an edge from the new node to any node in the existing graph ensures that the new graph G' is also minimally connected. Repeating this process for each additional node shows us that any fewer edges must result in a graph G which is disconnected, and that the minimum number of edges in the resulting graph is equal to $n-1$.

(b) The fewest number of edges such that any two nodes in G have a shortest path length of 1 would be one where each node is connected to every other node, also known as a maximally connected graph. This is also equal to $\sum_{i=1}^{n}(i-1)$, where n is the number of edges, or $\frac{n*(n-1)}{2}$. We can prove that this is the minimum through a proof by contradiction. Assume that there exists a graph G, where the number of edges such that any two nodes in G have a shortest path length of 1 is less than $\frac{n*(n-1)}{2}$. Then, there must exist at least one node for which the number of outgoing edges is less than the total number of nodes - 1 (we subtract the one to consider the fact that a node doesn't have an edge to itself). If this is the case, then this node is not connected to every node in the graph, which breaks the invariant stated above, namely that any two nodes in G have a shortest path length of 1.

(c) The fewest number of edges in this case would be $n-1$. Let us consider the simple $n = 3$ node case, where the maximum shortest path between any two nodes = 2. Such a graph can be described by X—Y—Z. From part (a) above, we know that the least number of edges for $n = 3$ is 2. If we wanted to add an additional node, we could want to connect that new node to Y, so that we can uphold our invariant where the shortest path length of ≤ 2. Since every node is connected to Y within a distance of 1, every node is connected to all other nodes, excluding Y, with a cost of 1+1 = 2. Once again, from part (a), we know that this is the fewest number of edges possible, since any fewer edges would lead to a disconnected graph.

□

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 does the average shortest path length seem to asymptotically approach and why?

Solution:

(a) (Only need to include implementation in graph.py.)

(b) (Only need to include implementation in graph.py.)

(c) (Include implementation in graph.py and execution trace in avg_shortest_path.txt.)
Final average length: 1.893.

(d) (Include implementation in graph.py, data in varying_p.txt, and graph as image file separately or in here.)

(e) As p increases, we asymptotically approach 1. This makes sense because with a probability of 1, every pair of nodes are ensured to have an edge between them with a cost of 1. The average shortest path length models the function $1/x + 1$.

□

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.

Solution:

(a) (Include implementation in graph.py and execution trace in fb_shortest_path.txt.)
Final average length: 3.752.

(b) **Probability p:** $\sim$ 0.0108. This can be calculated by dividing the total number of edges (88234) by the total number of possible edges = $n(n1)/2$, where n = 4039. Plugging this into the formula, we see that this is equal to 8154741.

(c) The average shortest path length of the Facebook data appears to be greater than expected if it were a random graph with the same number of nodes and value of p. For a random graph of (n = 4039, p = 0.0108), we had an average distance of $\sim$ 2.607.
Intuitively, one reason for why the average distance in the real world is greater could be due to geographical clusters. Nodes that are close to one another in a particular city in the West Coast are less likely to be connected to nodes located on the East Coast. This lower probability, in turn, increases the average distance of the network. In a random graph, however, every node has the same probability to be connected with any other node, which helps the network maintain a lower average distance.

□

```
from collections import deque
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

New Section

8A

In graph.py, implement (and turn in) a function create graph(n,p) that produces an undirected graph with n nodes wh
with probability p.

```
def create_graph(n, p):
  graph = dict()
  for i in range(n):
    for j in range(i+1, n):
      if (i not in graph.keys()):
        graph[i] = []
      if (j not in graph.keys()):
        graph[j] = []
      if (random.random() <= p):
        graph[i] = graph[i] + [j]
        graph[j] = graph[j] + [i]
  return graph
```

8B

Implement a general shortest-path algorithm for graphs, as described in lecture, that works on your graph. In graph.
outputs the length of the shortest path from node i to j in your graph G. Make sure to handle the case where the gr
by outputting “infinity”.

```
#Code implementation courtesy of --> https://codereview.stackexchange.com/questions/193410/

def shortest_path(G, i, j):
  if i == j:
    return [i]
  visited = {i}
  queue = deque([(i, [])])

  while queue:
    current, path = queue.popleft()
    visited.add(current)
    for neighbor in G[current]:
      if neighbor == j:
        return path + [current, neighbor]
      if neighbor in visited:
        continue
      queue.append((neighbor, path + [current]))
      visited.add(neighbor)
  return "infinity"  # no path found. not strictly needed
```

8C

Construct a graph for $n = 1000$ and $p = 0.1$. Estimate the average shortest path between a random pair of two (c
repeat for 1000 random pairs of nodes in your graph. Output an execution trace in **avgshortestpath.txt** containing a

```
avg_path_doc = open("/content/avg_shortest_path.txt", "w")
graph_1000 = create_graph(1000, 0.1)
lengths = []
for i in range(1000):
  rand_node_one = random.randint(0, 999)
  rand_node_two = random.randint(0, 999)
  z = shortest_path(graph_1000, rand_node_one, rand_node_two)
  lengths.append(len(z)-1)
  avg_path_doc.write("(%d, %d, %d)\n" %(rand_node_one, rand_node_two, len(z)-1))

avg_length = sum(lengths)/1000
print(avg_length)
```

```
1.898
```

8D

For $n = 1000$, run the shortest-path algorithm on data sets for many values of p (for instance, 0.01 to 0.04 using .0
increments). Turn in your numerical data as **varyingp.txt**, and plot the average shortest path as a function of p and s
extension) or include in **yourmain.pdf** file.

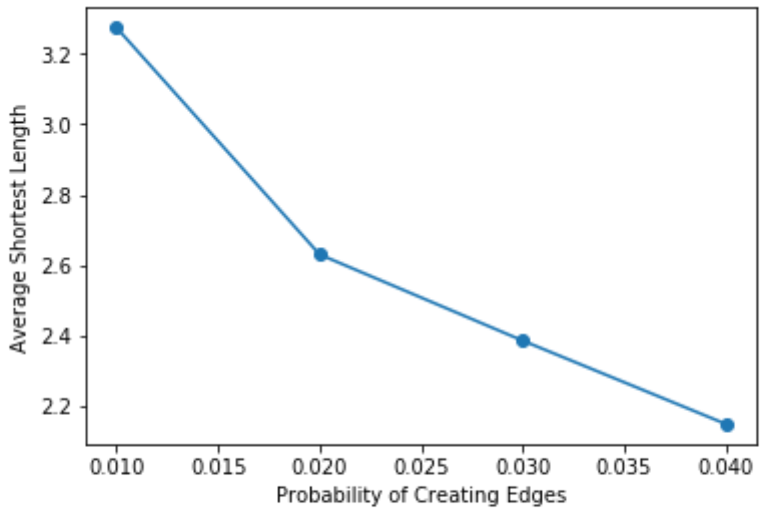
Note: For p= 0.01 there is actually a small but reasonable chance (around 4%) toproduce a disconnected graph. If this graphfor the purposes of gathering data.

```
avg_len = []
x_val = []

for i in np.arange(0.01, 0.05, 0.01):
  x_val.append(i)
  graph_varying_p = create_graph(1000, i)
  lengths = []
  for j in range(1000):
    rand_node_one = random.randint(0, 999)
    rand_node_two = random.randint(0, 999)
    z = shortest_path(graph_varying_p, rand_node_one, rand_node_two)
    lengths.append(len(z)-1)
  avg_len.append(sum(lengths)/1000)

plt.plot(x_val, avg_len, '-o')
plt.xlabel("Probability of Creating Edges")
plt.ylabel("Average Shortest Length")
```

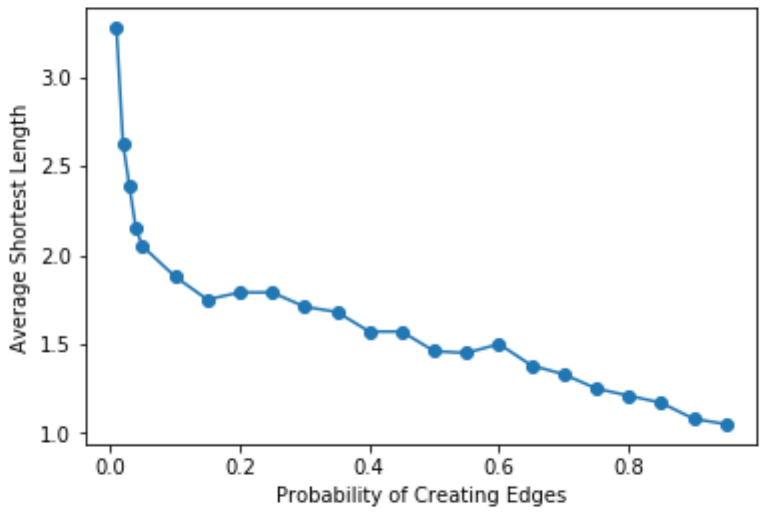
```
Text(0, 0.5, 'Average Shortest Length')
```



```
for i in np.arange(0.05, 1, 0.05):
  x_val.append(i)
  graph_varying_p = create_graph(1000, i)
  lengths = []
  for i in range(100):
    rand_node_one = random.randint(0, 999)
    rand_node_two = random.randint(0, 999)
    z = shortest_path(graph_varying_p, rand_node_one, rand_node_two)
    lengths.append(len(z)-1)
  avg_len.append(sum(lengths)/100)

plt.plot(x_val,avg_len, '-o')
plt.xlabel("Probability of Creating Edges")
plt.ylabel("Average Shortest Length")
```

```
Text(0, 0.5, 'Average Shortest Length')
```



```
#writing varying_p.csv for submission

varying_p_data = list(zip(x_val, avg_len))
varying_p_df = pd.DataFrame(varying_p_data, columns = ['Probability', ' Average Shortest Le

varying_p_df.index.name = "varying_p"
varying_p_df.to_csv("/content/varying_p.csv")
```

8E

Intuitively explain the behavior of the data you found; specifically, as p increases (in particular, look at the larger valu
average shortest path length seem to asymptotically approach and why?

The average shortest path length of the Facebook data appears to be greater than expected if it were a random gra
p. For a random graph of (n = 4039, p = 0.005), we had an average distance of ~ 3.04.

9(a) - (c)

```
fb_df = pd.read_csv("https://raw.githubusercontent.com/hui-liu/Bioinformatics-Scripts/maste
```

```
# getting data from csv into adjacency list

from collections import defaultdict

fb_nodes = set(fb_df.iloc[:,0])
fb_graph = defaultdict(set)

for i in range(len(fb_df)):
  fb_graph[fb_df.iloc[i,0]].add(fb_df.iloc[i,1])
  fb_graph[fb_df.iloc[i,1]].add(fb_df.iloc[i,0])
```

```
fb_df.iloc[:, 0]
```

```
fb_path_doc = open("/content/fb_shortest_path.txt", "w")

lengths = []
for i in range(1000):
  rand_node_one = random.randint(0, max(fb_nodes))
  rand_node_two = random.randint(0, max(fb_nodes))
  z = shortest_path(fb_graph, rand_node_one, rand_node_two)
  lengths.append(len(z)-1)
  fb_path_doc.write("(%d, %d, %d)\n" %(rand_node_one, rand_node_two, len(z)-1))

avg_len = (sum(lengths)/1000)
print(avg_len)
```

```
3.752
```

```
n = 4039
fb_edges = 88234
total_edges = n*(n-1)/2

prob = fb_edges/total_edges
prob
```

```
0.010819963503439287
```

```
max(fb_nodes)
```

```
4031
```

```
graph_fb = create_graph(4039, 0.0108)
lengths = []
for i in range(1000):
  rand_node_one = random.randint(0, 4038)
  rand_node_two = random.randint(0, 4038)
  z = shortest_path(graph_fb, rand_node_one, rand_node_two)
  lengths.append(len(z)-1)

avg_length = sum(lengths)/1000
print(avg_length)
```

```
2.607
```