4 Pandemaniac [80 points]

Welcome to our second mini-project! Here we will crown the "pandemaniac", i.e. the king/queen of pandemics.

This assignment is meant to give you a flavor of one of the "hot" topics in complex networks these days — competing cascades. We talked about cascades in lecture, but in that context we only had one new product which was looking to take over the network. Here, we'll consider a setting with multiple cascades competing. In fact, each group will be responsible for seeding a cascade with the goal of taking over as much of the network as possible, and we'll see which team wins!

Your Task: In groups of 2 or 3, create an algorithm that will take as input an undirected graph (in adjacency-list form) and output a list of N seed nodes where your epidemic will start (where N is given). You will be competing against other teams who are also trying to spread their epidemics on the same graph. Your team's goal is to take over more of the network (i.e., a larger number of nodes) with your epidemic than your competitors do with theirs.

The Epidemics:

As you can imagine, the key to this task is understanding the way the epidemic spreads. We'll be using a simple, deterministic model for the spread that is similar to what we studied in class. In particular, each node in the graph can be acquired by at most one team (color). Nodes start uncolored and then may be convinced to adopt a color based on the colors of their neighbors. After they adopt a color, they still may later be convinced to switch their color if their neighbors switch. We describe the process in detail below, and you can download the code to simulate the epidemic process at **Resources of Piazza website**

Seed nodes

To begin, the entire graph is uncolored at iteration 0. Each team is then allowed to pick a set of nodes – called seed nodes – to acquire in iteration 1. **Note that, if more than one team picks the same node, then none of those teams gets that node!** These seed nodes will then be colored accordingly.

The spread of the epidemics

From the seed nodes, the epidemics spread iteratively. During each iteration, every node chooses its next color through a majority vote among itself and its direct neighbors. All colored direct neighbors of the node vote for their respective color with 1 vote; however, if the node itself is currently colored, it votes for its own color with 1.5 votes. If any color has a **strict majority** out of the total number of votes, then that becomes the next color for the node.

If there is no strict majority, then the node keeps its current color or remains uncolored. These cases are demonstrated in Figures 3 and 4. Note that it is (slightly) harder to convert a node after it has been acquired, but nodes can switch colors multiple times.

Termination

The iteration spread of the epidemics will continue until it stabilizes, i.e. when no nodes of the graph change colors. Note that convergence is not guaranteed; however, cyclic behavior is unlikely. In the unlikely scenario in which the epidemics do not converge, we will terminate the simulation after a random number of iterations uniformly chosen between 100 and 200.

The Details:

(a) *Website:* Submissions are done via the Pandemaniac website at `http://pandemaniac.bkspin.`

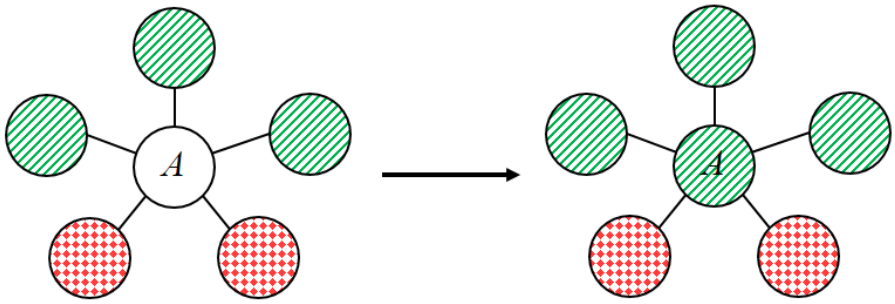

Figure 3: An uncolored node *A* converting to "striped" (since "spotted" gets 2 votes and "striped" gets 3 votes, which is a majority among the 5 votes).

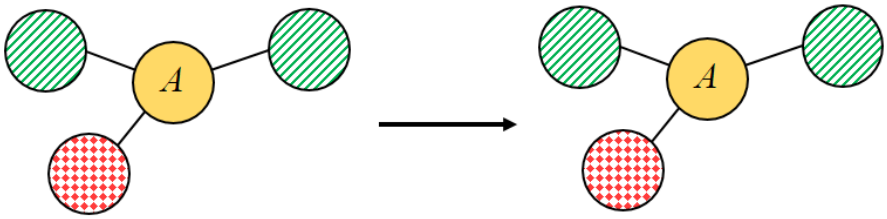

Figure 4: Node *A* doesn't change color because no color got the majority of votes ("striped" gets 2 votes, "dotted" gets 1 vote, "solid" gets 1.5 votes, none of them gets a majority among the 4.5 votes).

com. Please follow the instructions on the website on how to register a team, where to make submissions, and where to view the results.

(b) *Graphs & Runs:* We will have a "regular season" during the week of the assignment and then a "tournament" at the end of the week. In particular, there will be several graphs to practice on over a period of 5 days prior to the tournament. During the regular season, each graph will have 2 to 8 randomly-matched teams competing against each other.

The graphs are named in the following (semantic) way:

num_players . *num_seeds* . *unique_id*

- *num_players* is an integer representing the number of players that will compete on this graph,
- *num_seeds* is an integer representing the number of seed nodes that your strategy must select,
- *unique_id* is an integer that has no significance except to distinguish between different graphs of the same type.

You must create a program that will accept as input an undirected graph in adjacency-list form via a JSON file. This file contains a JSON object with (string) keys representing the nodes and list of string values representing the direct neighbors. Note that if a node has no neighbors (albeit unlikely), then it is still present as a key in the JSON object, but with an empty array as a value. You may assume that the nodes of the graph are numeric strings. If there are V nodes, then they are numbered from 0 to $V-1$.

For example, the graph of a square, with vertices labeled 0 through 3 in a clockwise fashion, will be represented by the following:

```
{
  "0": ["1", "3"],
  "1": ["0", "2"],
  "2": ["1", "3"],
  "3": ["0", "2"]
}
```

(c) *Submissions:* When your team is ready, you may download the graph, after which you will have a set amount of time (3-5 minutes, depending on the graph, **TIME LIMIT!**) to run your program and choose a set of seed nodes. Your team will then upload a file containing your list of seed nodes. **Each line of the file must contain only one node**. Since we are running **50 rounds of game for each graph** in order to reduce the variability of the result, you need to submit $[\mathbf{num\_seeds} * \mathbf{50}]$ seed nodes. For example, a submission to a graph named `1.4.4` (i.e. a graph that requires choosing 4 seeds) could be:

$$
\begin{aligned}
1^{\text{st}}\ \text{round} &\begin{cases} 0 \\ 1 \\ 2 \\ 3 \end{cases} \\
2^{\text{nd}}\ \text{round} &\begin{cases} 0 \\ 1 \\ 2 \\ 3 \end{cases} \\
&\vdots \\
50^{\text{th}}\ \text{round} &\begin{cases} 0 \\ 1 \\ 2 \\ 3 \end{cases}
\end{aligned}
$$

There should be total $4 * 50 = 200$ nodes in the text file. Every 4 nodes in your submission will be used to run the i^{th} round of the game for each graph. You can make multiple submissions before your time is up. Once the timer expires, we will use your most recent successful submission.

(d) *The Regular Season:* The regular season matches will use 10 graphs every day from **February 20-24 (Friday to Tuesday).**. These graphs will change each day. In each of these graphs, you will play against 1 to 7 other randomly-chosen teams. (If there aren't enough teams to compete, then there will be filler teams who pick random seed nodes). Submissions for each day's practice rounds will close at **11:59pm** (beginning February 20). The results will be available on the website by 9:00am the following day.

(e) *TA Graphs:* The TAs will also participate in the regular season and will compete 1-on-1 with each team on specially designated graphs. These graphs will all be named `2.10.X`, where `X` is an arbitrary number. More specifically, the TA graphs for each day are:

Day	TA-degree	TA-fewer	TA-more
Feb 20	`2.10.10`	`2.10.20`	`2.10.30`
Feb 21	`2.10.11`	`2.10.21`	`2.10.31`
Feb 22	`2.10.12`	`2.10.22`	`2.10.32`
Feb 23	`2.10.13`	`2.10.23`	`2.10.33`
Feb 24	`2.10.14`	`2.10.24`	`2.10.34`

It is **highly recommended** you participate in the regular season in order to (1) get points for beating the TA teams and (2) practice and develop your strategy against other teams so that you can win the actual competition! The details of the strategies used by the TA teams are given in the grading section below.

(f) *The Tournament:* The tournament will last 3 days, starting from **Wednesday, February 25**. For each day, there will be 3 or 4 stages (depending on the number of participating teams), each consisting of 3 games (with 3 different graphs, of course). At the end of each stage, half of the teams – those with the lowest average score in the 3 games of the stage – will be eliminated from the pool. Teams that tie at the threshold will also be eliminated. Results for each stage of the competition will be announced on the following day. Your team must submit your seed nodes for the competition graphs by **11:59pm** on each competition day. As before, once you download a graph, you will have a limited amount of time to submit your seed nodes for that graph. After you make your last submission, your algorithm (and code) is final and may not be modified anymore.

During the tournament, each game runs for 50 rounds as described above. Points are awarded in each round of each stage based on a team's rank. If two teams tie, they will get the same number of points. Rankings will follow standard competition ranking (for example, if two teams tie for second, the places are 1st, 2nd, 2nd, 4th). Points will be awarded according to the following table:

Place	1st	2nd	3rd	4th	5th	6th	7th	8th	9th and beyond
Points	20	15	12	9	6	4	2	1	0

The final total tournament score is calculated using the following formula:

$$P_{final} = \frac{1}{6}(1 \times P_{day1} + 2 \times P_{day2} + 3 \times P_{day3})$$

which means that even if your performance is not that good during the first two days, you'll still be able to change the situation since day 3 is weighted more!

Grading:

- **Report [50 points]:** The most important part of this assignment is the thought that goes into your algorithm. So, the bulk of your grade comes from a report detailing your design process. Specifically, your team must submit one detailed report describing your strategy and the reasoning behind it. It must also describe the contributions of each team member as well as citations to any papers or other resources that helped in the formulation of your strategy.

  A good report will show that you put significant thought into whether and how you could use different centrality measures, clustering measures, and game theoretic principles. We also ask that you include suggestions on how we could change/improve it for next year! **This report, along with your final code, is due 5pm, Friday, February 28, by email to the TAs (cs144caltech2015@gmail.com)**.

- **Counterexample [5 points]:** Give a graph and an initial coloring such that the epidemic colors do not converge. Include this in your report.

- **Start early! [5 points]** Participate in the first practice round on **February 20**.

- **Beat the TAs [20 points] :** The TAs will participate in the practice rounds every day and will compete 1 on 1 with each team on specially-designated graphs. You will receive 10 points for each TA team that you beat (and only have to beat each TA team for one 50-round game):

  – Compete against the TA who gets fewer seed nodes than you do. (**10 Points**)

  - Compete against the TA who picks the nodes with the highest degree. (**10 Points**)
  - Compete against the TA who gets 20% more seed nodes than you do. (**Optional, 5 Bonus Points**)

- **Class leaderboard [Bragging Rights]:** At the end of the project, at 9:00am on March 1, the team with the best score will be crowned "Pandemaniacs", and be listed in perpetuum on the course website.

Hints and Notes:

- Feel free to use any libraries that may help you. The `json` library in Python is very useful for parsing JSON. You have also created some algorithms in Homeworks 1 and 2 that may be of use!
- A natural idea for your algorithm is to pick seed nodes that can easily spread your epidemic to other nodes. Using centrality measures might be a good way to pick these nodes, but which centrality measures are best? Also, be sure to take into account the fact that other teams might be using the same strategy as you (in which case your seed nodes will cancel out).
- Don't forget about clustering. Remember from class that clusters provide a key predictor to how far a cascade will spread.
- You only have a short amount of time to pick seed nodes, so your algorithm must run within that time. However, do you *really* need to examine all nodes? Is there any other information you could use about the graph to approximate your calculations?
- We are providing you the epidemic simulation script that will allow you to run the simulations on your own. This can be useful for tuning your algorithms, but it could also be part of the algorithm itself! Could you calculate the effectiveness of competing strategies by pitting them against each other? Could a game tree be useful?