**I155 Final Project**

Code due Mon Aug 12, Presentations begin Wed, Aug 14

**Social Graph Simulation and Analysis**

In this project, you will simulate competition between several different technologies in a population of users. Each user has connections to some number of other users, forming a social graph. At the beginning, each new technology is used by a small number of users called first adopters. Over time, the technologies spread along the graph connections, as each user adopts a technology recommended by their peers.

Your programming objectives are as follows:

**1. Graph Construction.** You will need to write three classes, a Graph class, representing the entire population of users, a User class, representing a single person in the graph, and a Technology class, representing one company's product. These classes are outlined with a number of method headers in the final project [framework code](http://courses.ischool.berkeley.edu/i155/su13/code/final_framework.py). You should fill in these methods and add more attributes as necessary to complete the simulation.

The graph constructor takes a population as a parameter and creates new set of users, numbered 0, 1,..., population - 1. The number of a user should be returned by its get\_id() method. When a new graph is constructed, there should be no connections between users and no users should have any technology.

You should write two methods to add connections to your new graph. The circle\_connect(n) method should connect each user i to the next n users: i + 1, i + 2,.. i + n, where the sums are mod population (looping back to user 0 to form a circle). This is an important method because it guarrantees that the graph is connected, and a technology can eventually reach every user. On the other hand, the random\_connections() method should add the given number of random connections between users. Make sure that you model each connection as symmetric. If user a is a friend of user b, user b is also a friend of user a.

Your Graph class should also include an is\_connected() method, which should return True if there is a path from one user to every other user along the graph connections. Your method will need to follow paths through the graph, which may be done iteratively or recursively.

**2. Time Simulation.** You will run what is known as a discrete time simulation on your graph. This means that the model has rules that govern how the state changes from one time period to the next. Before the simulation begins, select a small number of agents to be first adopters of each technology. In each time period, each user surveys the technologies in use by themselves and all of their friends. The user then adopts the technology that's most popular in this subset of users. This becomes the user's technology for the next time period. If there are multiple technologies that tie for most popular, the user selects one at random. Finally, if no technology is in use by either the user or their friends, the user does not adopt any technology.

Warning: A common mistake in this type of simulation is in letting the technology pass through multiple connections in one time period. Suppose user 0 has a technology at time 0. User 0 is connected to user 1, and user 1 is connected to user 2. If your program processes user 0 first, she might pass the technology to user 1. If your program then looks at user 1 and sees that he has the new technology, it might allow user 1 to immediately pass the new technology to user 2, so all three users might have the technology at time 1. That should not be allowed to happen, because the new technology should only move through one connection in each time period. To prevent this situation, you must carefully separate one period from the next. User 0 has the new technology at time 0, so if she gives it to user 1, user 1 will have the technology at time 1, and user 2 can only get the new technology at time 2.

To assist with grading, please use the function headers given in the [framework code](http://courses.ischool.berkeley.edu/i155/su13/code/final_framework.py).

**3. Graph Analysis.** In the final part of the project, you will play the role of a company that has created a new technology and hopes for it to reach as many users as possible. You will write a new class, GraphAnalyzer, which contains code to examine a graph. The GraphAnalyzer constructor should take two parameters: a Graph to study and a Technology, representing the technology your company is trying to promote. The most important method in GraphAnalyzer is choose\_user. Every time this method is called, it should return a new user to be a first adopter of the company's technology. You may imagine that the company targets these users with special promotions to ensure that they have the new technology at time 0. The idea behind this method is to choose users so that the technology spreads through the graph, reaching more users than the competing companies.

To write your choose\_user method, you will have to decide on a strategy for selecting users. You may consider, for example, the ids of the users, how many connections they have, the degrees of separation between users, and so on. Your method may be as complex as you like, however it must finish running within 30 seconds on a basic laptop. Be careful, as many algorithms that explore paths through a graph take exponential time as the length of the path increases.

There are many possibly strategies you could use in your choose\_user method, but we will see which team has the best algorithm with a final class competition.

For the class competition, each team will play the role of a competing company with a new technlology. We will create a number of graphs and use each team's GraphAnalyzer to choose that team's first adopters. We will then run the graph simulation for a large number of periods, say 100. One company may take over the entire graph by this point, or multiple companies may be deadlocked, none able to dislodge its competitors.

Note that we will be using our own Graph, User, and Technology classes to generate graphs for this competition. You must write your GraphAnalyzer so that it only calls the Graph and User methods provided in the [framework code](http://courses.ischool.berkeley.edu/i155/su13/code/final_framework.py). Do not add another method to Graph, User, or Technology and then call this method from within GraphAnalyzer, since the method may not exist in the competition objects.

For this competition, we will use graphs of 500 users. We will call circle\_connect(3), then add 50 random connections. The teams' GraphAnalyzers will be arranged in random order. We will then call choose\_user on each GraphAnalyzer in this order, allowing each team to select one user at a time. We will repeat this process three times, so that each team begins the simulation with three users.

Note: You should make sure that your choose\_user method only returns users that do not already have a technology. Any user that already has a technology will be ignored, and your team will begin the simulation with fewer first adopters.

The winner of each simulation is the team with the most users at the end. Points will be awarded for top-place finishers, and these points will be tallied across a large number of simulations.

**Deliverables**

1. You should prepare a 5-10 minute presentation for the class. You may demonstrate a run of your simulation, discuss your strategy for choose\_user, or present any other parts of your code that are interesting.

2. Please place your script in a single file, final\_team\_i.py, where i is your team number. Comment out your main script so that only the classes are defined when your code is run, and email it to zschneider at berkeley.edu.