MATH.APP.270 Algorithms for graphs

Programming assignment 3 2023

This programming assignment deals with analyzing certain probabilities associated with graphs (or digraphs). You must choose one of two games, whose details will be explained later.

The input for both games is a digraph that is meant to model some social network. The size of input digraphs vary, but it is safe to assume that there will never be more than 3000 vertices. The input digraph may be weighted or unweighted.

The 'game' will be played with the digraph. The vertices are the players in the game. Edge (u, v) exists when there is some relationship between players u and v. The code that you should submit should analyze the digraph structure and produce the winner (or winners) of the game.

Your two choices for the game are as follows:

1. Pass-the-baton (Kapularalli)
2. Tattletale (Juorukello)

Details of these two games are given on the next pages. Both games use a Monte Carlo simulation approach. (See `https://en.wikipedia.org/wiki/Monte_Carlo_method`.) The Monte Carlo simulators for the two games are provided on the Moodle page in the following two Python files: `MCKapularalli.py` and `MCJuorukello.py`.

You may use some language other than Python, but if you do you must also meet the following requirements:

- Your code should accept as an input a weighted digraph that is stored in the form specified on the course Moodle page.
- You must provide clear and detailed instructions on how to compile your code and how to run your code. You cannot assume that a user has any particular knowledge. For example, you should not assume that user knows anything about Java or how to use Netbeans.

Pass-the-baton (Kapularalli) This game starts by randomly selecting a player (a vertex) who gets the baton. Call this starting player x. Of all the players related to x, x selects one randomly and passes the baton to her/him. This new player in turn passes the baton and so on. Passing of the baton is repeated thousands of times.

If the digraph is weighted, the probability that player x passes the baton to player v is given by the following formula:

$$p(v) = \frac{w(x,v)}{\sum_t w(x,t)} .$$

Each time the baton is to be passed there is a small probability that it will be given to some randomly choosen player. This can be any player, not necessarily a player related to the player who currently holds the baton. This small probability is given by parameter d, which is specified at the start of the game.

Passing of the baton is repeated until the 'music' stops. The player who has the baton when the music stops earns a point. You may assume that the baton gets passed thousands of times before the music stops. The game itself is played numerous times so that it is relatively safe to assume that each player will earn some points. Assuming the game is played N times, where N is some very large number, each player will have earned some portion of the N points. Your task is to sort the players in decreasing order based on the number of points you expect them to earn after N games have been played.

Code specification Your code should be a function that takes as input a digraph and a parameter d. Your function should return a list of the players. The players in the list should be ordered in terms of the number of points you would expect them to receive when Pass-the-baton is repeated thousands or millions of times starting from the largest point earner and ending with the smallest point earner. Your function should not return the points each player gets. Also the order of the players should not be based solely on the Monte Carlo simulations. Rather the order of the players should be based on the digraph structure.

In each case the players will be numbered 1, 2, ... n. Hence your function should return some permutation of these integers. The first player in your permutations corresponds to the player who you expect would earn the most points, the second corresponds to the player who you expect would earn the second most points and so on.

Tattletale (Juorukello) This game is deals with the passing of a 'tale' or some piece of gossip from one player to the next. This tale itself is about one particular player, who will be called the *subject* of the tale. The game starts with one player, call him/her x_1, being randomly selected and being told a tale subject y. The subject y is also randomly selected from all players. Player x_1 tells the tale to all players to whom she/he is related. (The speed with which x_1 gets to tell the tale may be affected by the edge weight as explained below.) Each player who hears the tale, tells it to all player to whom they are related. Each time a player tells the tale to another player, they add the phrase 'I heard from player x_i ' to the start of the tale. For example, if player x_3 heard the tale from player x_2 who heard the tale from player x_1, then when player x_3 tells the tale, player x_3 will start by saying 'I heard from x_2, who heard from x_1, that subject y ...'.

If the digraph modeling the game is weighted, then the weight $w(x_i, x_j)$ can be interpreted as the number of hours it takes for player x_i to get to player x_j to tell the tale. If the digraph is unweighted, then you may assume the weight of each edge is 1.

You may assume that only those tales told along the shortest paths are significant. Hence if player x_i first hears the tale via some shortest path and then has the possibility to hear the tale again via some other path that is longer than the shortest path, this second (longer) path should be ignored. All shortest paths should equal probability of being used.

When the tale's subject y finally hears the tale, the telling of the tale stops. Each player along the path via which the tale got passed from x_1 to y receives one point, except for y and x_1.

The Tattletale game is repeated N times, where N is a very large number.

Code specification Your code should be a function that takes as input a digraph. Your function should return a list of the players. The players in the list should be ordered in terms of the number of points you would expect them to receive when Tattletale is repeated thousands or millions of times starting from the largest point earner and ending with the smallest point earner. Your function should not return the points each player gets. Also the order of the players should not be based solely on the Monte Carlo simulations. Rather the order of the players should be based on the digraph structure.

When interpreted as an undirected graph, you may assume that the input is a connected graph. The digraph itself may not be strongly connected, but this is not significant. If no path exists from the first player to the subject, then no player gets points for that game. (Of course the code must detect that such a path does not exist.)