Broadcasting a message within a Social Network

Overview

This project investigates the problem of making an announcement which I want to reach everybody within a social network. How many people do I need to announce that? I just want to find a set of people to essentially announce to the social network that something's true, and everyone will hear it. Ideally I would like to find the minimum set of people, but for this first part of the project it won't be always the case.

Data

First I will use some tiny data sample in order to test the algorithm and check it will be working fine, afterward I will test the algorithm with a huge data sample, the provided USCD Facebook data. The Facebook data represents friendships between students at USCD in 2005. Every line in the file specifies a particular friendship relationship between two users, just numbers, which are the user IDs.

So, for example, the first line in this file says that user 0 is friends with user 22. It would be an undirected graph, because the friendship has to exist in both directions, but the algorithm will create a directed graph.

Questions

I want to find the group of people highly connected, so with this group I can reach everyone in the network. This will be a dominating set.

Algorithms, Data Structures and answer to your question

Main Data Structure: The network has been laid out as a classic graph using an adjacency list. Each individual in the graph is a vertex and an edge between vertices represents a relationship. In order to represent the Graph I use a map, each entry in the map is a vertex(key) and the value is a list with its neighbors.

Algorithm. I will try a greedy algorithm that is described in (Johnson, 1973). Basically, at each stage, the greedy algorithm chooses the set which contains the largest number of uncovered elements. Starting from an empty set, if the current sub-set of vertices is not the dominating set, a new vertex which has the most number of the ad-jacent vertices that are not adjacent to any vertex in the current set will be added.

The greedy algorithm would work as follow:

- Mark all vértices as “uncovered”
- Initialize an empty dominant set
- While there are uncovered vértices:

- Find the vertex, v, which would cover the most uncovered vértices
- Add v to the dominant set
- Mark that vertex and all of its uncovered neighbors as covered

Algorithm Analysis

Finding a Dominant Set the Greedy approximate algorithm takes a polynomial time, the greedy algorithm provides a Log(n) time complexity (Source: http://math.mit.edu/~goemans/18434S06/setcover-tamara.pdf).

In my particular example:

Getting a set with all the vértices in the graph: O(1)

The algorithm iterates over the set of vértices. In the first iteration it takes the node that has more neighbors, adds this node to the dominant set, and delete from the set of vértices the node and its neighbors. The algorithm repeats the process until the set of vértices is empty.

The algorithm takes O(Log(n)) time complexity.

Correctness verification

In order to test my implementation of the algorithm I developed a JUnit class to test the basic cases, toy examples of graphs with a few vértices and edges, and in all cases it worked as I expected.

I developed a static main method in order to check the behavior of the algorithm with a big data example (Facebook data).

Reflection

The Greedy algorithm worked in each case as I expected, it is a very well known and tested algorithm. In some cases it found out the mínimum dominant set, but in other cases only a dominant set.