Complex Systems 535/Physics 508: Homework 3

1. What (roughly) is the time complexity of:

   (i) Vacuuming a carpet if the size of the input to the operation is the number n of square feet of carpet?

   (ii) Finding a word in a (paper) dictionary if the size of the input is the number n of words in the dictionary?

   Explain your reasoning in each case.

2. For a network of n vertices show that:

   (i) It takes time $\mathrm{O}(n^2)$ to multiply the adjacency matrix into an arbitrary vector if the network is stored in adjacency matrix form, but only time $\mathrm{O}(m)$ if it is in adjacency list form.

   (ii) It takes time $\mathrm{O}(n(n+m))$ to find the diameter of the network.

   (iii) It takes time $\mathrm{O}(\langle k \rangle)$ to list the neighbors of a vertex, on average, but time $\mathrm{O}(\langle k^2 \rangle)$ to list the second neighbors. You can assume the network is stored in adjacency list format. (In a network with a power-law degree distribution where $\langle k \rangle$ is finite but $\langle k^2 \rangle$ formally diverges, this means the second operation is much more work than the first.)

   (iv) For a directed network in which in- and out-degrees are uncorrelated, it takes time $\mathrm{O}(m^2/n)$ to calculate the reciprocity of the network. Why is the restriction to uncorrelated degrees necessary? What could happen if they were correlated?

Continued over the page →

3. Here are the results of the in-class social network questionnaire:

Number	Name	People they know
1	Agrawal, Mayank	16, 23
2	Allington, Noah	27
3	Cai, Jiarui	16, 18
4	Cale, Mario	
5	Calloway, Ian	
6	Cantwell, George	15
7	Cheng, Frank	
8	Choe, Ho	
9	Columna, Andres	
10	Chockalingam, Valliappa	
11	Foakes, Gregory	
12	Guo, Tong	21, 27, 33
13	Hudson, Aristos	
14	Kim, Yura	18
15	Klishin, Andrei	6, 20, 23, 24
16	Krishnan, Rashmi	1, 3, 23
17	Lin, Sunny	22
18	Liu, Boang	14, 22
19	Macdonald, Christian	
20	Shields, Colin	9, 11, 23
21	Song, Shiya	12, 27, 33
22	Sun, Ruoyan	13, 17, 18
23	Thomas, Albert	1, 16, 24
24	Varghese, Arun	15, 23, 32
25	Wang, Chris	
26	Wu, Jiaxing	27, 29
27	Ye, Teng	2, 12, 21, 26, 33
28	Zhang, Pengbo	
29	Zhang, Rui	
30	Zhang, Shang	15, 26
31	Zhang, Xinyu	
32	Zhao, Xinyan	
33	Zou, Zhengting	12, 21, 27

You can also download the same data as a computer file from:

`http://umich.edu/~mejn/courses/2015/cscs535/classnet.txt`

Note that the network is directed, e.g., node 3 knows node 18 but not *vice versa*. Let us define an undirected *symmetrized network*, in which there is an undirected edge between nodes i and j if there is a directed edge either from i to j *or* from j to i.

To complete this homework problem you'll need to use network analysis software, so your first task is to download and install the network analysis software *Gephi* (`www.gephi.org`, available for PC, Mac, or Linux). You second task is to work out a way to turn the raw network data into a symmetrized undirected network and load it into Gephi. You can enter networks into Gephi by hand, but the program can also read network data in many file formats, so you may find it easier to convert the data file into a form Gephi understands and then read it in directly. (When I did it I used Emacs macros to turn the raw data into a simple adjacency list form, then wrote a short program to convert that to the GML file format and then loaded the result into Gephi. But everyone has their own way of doing these things.)

(i) When you load up the network the program will produce a picture of it, but the default picture is not very clear. Work out how to do a clear visualization. (Hint: Try out the different vizualization algorithms and see what works best for you.) Work out how to zoom in and out of the picture to examine different parts of the network. Work out how to turn on the node labels, so you can see which nodes correspond to which individuals. You may also need to change the size of the labels. (Hint: Look at the buttons and sliders along the bottom and sides of the network.)

(ii) What are the numbers n and m of vertices and edges network? Hence what is the average degree?

(iii) Calculate the degrees of all the nodes and make a histogram of the degree distribution.

(iv) What fraction of the network is occupied by the largest component? What is the diameter of the largest component?

(v) Calculate the eigenvector centrality of every node. Work out how to color the nodes on the screen according to eigenvector centrality so you can see which have the highest score. (Hint: the "Nodes" panel on the left is a good place to start.) Save a copy of the resulting visualization. Which three people in class have the highest eigenvector centralities?

(vi) Find the "communities" in the network using the modularity maximization method and make a picture of the network with node colors representing the communities. Roughly speaking, what is the computer doing when it performs this calculation? (We will look at community detection in networks in more detail later in the course.)

(vii) Do one other cool thing with this network using Gephi—calculate some interesting thing, or produce some interesting visualization. Explain what you did.

For full credit turn in your answers to the questions in parts (ii), (iv), (v), and (vi), your plots from parts (iii), (v), and (vi), and your results and explanation from (vii).

Continued over the page →

4. **Extra credit programming challenge:** This question is optional. You can get 100% on this week's homework without doing it (but you'll get extra credit if you do it).

   The file `http://umich.edu/~mejn/courses/2015/cscs535/internet.txt` contains a snapshot of the structure of the Internet at the autonomous system level in adjacency list format. The $n = 22\,963$ vertices in the network are numbered in order from zero (note: not from 1) up to 22962, with one line of the file for each one. A single line of the file looks, for example, like this:

   ```
   112 3 12 22 24
   ```

   This means that vertex 112 has degree 3, and its three neighbors are the vertices numbered 12, 22, and 24.

   Write a program in the programming language of your choice to do the following:

   (i) Read the data from the file into a data structure of your choice, to represent the network, in adjacency list format, in the memory of the computer. For example, in C you might use a dynamically allocated 2D integer array with a line for each vertex, and another 1D array for the degrees. In Python you might use an array of n lists or sets to store the neighbors of each vertex, and an array of integers for the degrees.

   (ii) Write code to calculate, using breadth-first search, the shortest distance from a given single vertex to every other, then average those distances to calculate the closeness centrality of the given vertex.

   Use your program to calculate the closeness centrality of the vertex numbered 0 in the network.

   For full extra credit, turn in a complete printout of your program, and a printout of it in action, showing the answer it finds.

   (Note: There are programs or libraries that you could download from the Internet that will do the closeness calculation for you. While it is perfectly legitimate under other circumstances to make use of such programs, doing so will not get you any credit on this question. You have to write your own breadth-first search program to get credit on this question.)