Network Analysis: Assignment

Please submit your solutions via duo by 2pm on 9 November 2018 as a zipped archive. Written answers should be in a single pdf document. At the top of the first page of this document identify yourself.

You must reference any sources used (other than the lecture notes). You can use any programming language you choose. Any code you write should be submitted but will not be directly assessed. You may reuse any code in the public domain but should clearly document what you have done. You may reuse code from the exercises done during the course (available on duo) without comment.

Some tasks are for level 4 students only. These are written **in bold**.

The marks available for correct answers to each question are indicated. Partial credit will be given for good attempts.

Your marked work will be returned no later than 7 December 2018.

You should make all graphs in this assignment **undirected**.

For each question you should report on what you did and include, as needed, illustrative plots. There is no expectation that you have access to any specialist computing resources, but you should aim, where possible, to illustrate relationships as clearly as possible. For example, recall that in Lecture 2 we obtained `na2random.png` to show the degree distribution of a random graph, but that in Lecture 3 we found that with a little more effort we could obtain `na3random.png`. In contrast, there may be problems where you must investigate a relationship by considering only a random sample of the vertices (if that is all your computational power allows). As a very rough rule of thumb, I would not expect any individual plot takes more than a couple of minutes to obtain on a university computer.

Question 1

[30 marks]

A *Ring Group Graph* is defined by four parameters m, k, p and q and is constructed as follows:

- Create mk vertices. The vertices are partitioned into m groups each of size k. The groups are labelled from 0 to $m-1$.
- For each pair of distinct vertices u and v, let ℓ_u and ℓ_v be the labels of the groups that u and v belong to respectively and
  - if $\ell_u = \ell_v$ (that is, u and v are in the same group) or if $|\ell_u - \ell_v| = 1 \bmod m$ (that is, u and v are in *adjacent* groups), add an edge between u and v with probability p,
  - otherwise add an edge between u and v with probability q.

Investigate the degree distribution of Ring Group Graphs for $p+q=0.5$, $p>q$. Decide which values of m, k, p and q to investigate. You should report on how the structure changes as p and q vary and whether the same effects are found for different values of m and k. Use plots to illustrate your observations. Investigate the relationship between the diameter of Ring Group Graphs and p (for fixed q, $p>q$). **Level 4 students: Investigate the relationship between the clustering coefficient of Ring Group Graphs and p (for fixed q, $p>q$).**

Question 2

[30 marks]

Construct the undirected graph defined in `coauthorship.txt`. Ignore edge weights.

A k-star is a set of $k+1$ vertices $\{0,1,2,\ldots,k\}$ such that 0 is joined to every other vertex and no other pair of vertices is adjacent; the centre 0 is the *centre* of the k-star. The *brilliance* of a vertex v, denoted $b(v)$, is the largest value k such that v is the centre of a k-star.

For the graph from `coauthorship.txt`, investigate the distribution of vertex brilliance.

For each of the following types of graph

- PA Graphs (see Lecture 3), and
- Ring Group Graphs (see Question 1 above)

create examples with approximately the same number of vertices and edges as the graph from `coauthorship.txt` and investigate the distribution of vertex brilliance. Comment on what you find.

Question 3 [40 marks]

This question is about the problem of *searching* in graphs. Note that the definitions here, particularly how the cost of a search is measured, differ from those you can find in the lectures or elsewhere.

The input to the search problem is a *start* vertex and a *target* vertex of a graph. The aim is to reach the target vertex from the start vertex by moving along the edges of the graph. The only information available at each vertex v is its id (see below) and the number $d(v)$ of its neighbours. The ids of the neighbours are stored in an array $N(v)$ and when the search reaches v, you can *query* the array and find $N(v)[i]$ for some i, $0 \leq i \leq N(v) - 1$. After making each query, either the search moves along the edge to one of the neighbours that has been queried, or another query is made (that is, you can recall all queries made so far at the current vertex). After $d(v)$ queries, when the ids of all neighbours are known, the search *must* move to one of the neighbours. The vertices visited during the search are not recorded and if a vertex is returned to, you must again query its array of neighbours (past queries are not now recalled).

The *search time* for a given start and target is the total number of queries made in reaching the target from the start (the number of intermediate vertices is not counted). The *search time* for a graph is the average search time over all pairs of start and target vertices. Clearly these times depend on the algorithm used for deciding when to query, when to move to a new vertex and how to decide which vertex to move to. The aim is that the search time is as small as possible.

For each of the following types of graph

- Random Graphs (see Lecture 2),
- Ring Group Graphs (see Question 1 above), and
- **Level 4 students: PA Graphs (see Lecture 3),**

describe an algorithm for searching in the graphs. (I expect that you will have different algorithms for the different types of graphs.) The ids of vertices are as follows:

- for Random Graphs, each vertex is labelled with a unique integer between 1 and n,
- for Ring Group Graphs, each vertex is assigned a unique integer and the label of its group (see Question 1),
- for PA Graphs, each vertex is labelled with a unique integer between 1 and n, and vertices $1, \ldots, m$ form the complete graph formed initially and then vertices $m + 1, \ldots, n$ are added in that order.

Note that the array of neighbours of a vertex should not be considered ordered. If you create graphs in such a way that the neighbours are ordered, then either randomize the ordering or, when querying the array of neighbours, choose a random member of the array (from amongst those not already queried).

You should explain why you believe your strategy might be effective and implement and test it on many instances. You can choose the parameters yourself as long as you are not perverse — for example, the groups in the Ring Group Graph should not be of size 1 or n — and it is acceptable that all your testing for a particular type of graph is on instances generated with the same parameters. As searching on graphs that are not connected can be impossible, you should choose parameters so that the graphs are very likely to be connected. Plot search time against the number of instances that achieve that time. Comment on your plots. It is acceptable to estimate search time by looking at only a sample set of pairs of vertices. Credit will be given for the effectiveness of the algorithm you design, and also, independently, for your explanation of the rationale behind the design.