CS319 – Scientific Computing

Lab 7: Networks and ranking

Goal: you will develop further proficiency with file input/output, and in using the Power Method for estimating eigenvectors.

1 PageRank

As we discussed Week 9 lectures, Google's fame and fortune was originally derived from their PageRank algorithm that gives an objective way of computing the relative importance of web-pages.

The basic idea is this: *the importance of a web-page is the probability that you are looking at it at any given time*. This idea has been generalised in many ways, from ranking sports teams, to finding the most influential people on Twitter.

Here we'll just consider an arbitrary network, and see which vertex is the most important, in some sense. Consider the following example:

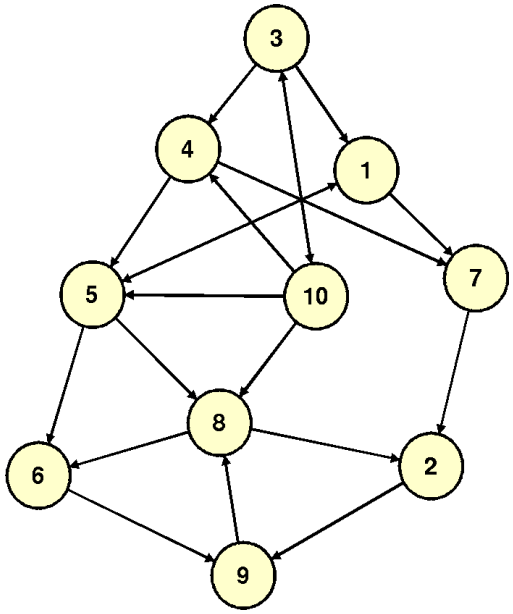

Figure 1: The graph for User 123

The generate a ranking for the nodes, we would:

1. Form the *adjacency matrix*, $A = (a_{ij})$, for the network: A is an $N \times N$ matrix, where N is the number of nodes in the graph, and

$$a_{i,j} = \begin{cases} 1 & \text{if there is a link from Node } i \text{ to Node } j \\ 0 & \text{otherwise} \end{cases}$$

2. Make the associated *Markov matrix*, $S = (s_{ij})$, where $s_{i,j}$ is the *proportion* of vertices in A which start at i and go to j. (That is, divide the entries in row i by the sum of the entries in that row). If there are no entries in a given column of A, set the corresponding entries of S to $1/N$.

3. Choose a "damping" value σ, e.g., $\sigma = 0.85$.

4. Set the matrix G to be $g_{ji} = \sigma s_{ij} + (1-\sigma)/N$. *Note that G is the transpose of $\sigma S + (1-\sigma)R$, where R is the matrix all of whose entries are $1/N$.*

5. Now find the eigenvector associated with the eigenvalue that is 1. For details of that part, see the notes from Week 9.

2 Your network

Download the file MakeGraphs.cpp from the "lab 7" section of the course website. Create a new project that includes this file, and the versions of `Matrix09.h` `Vector09.h` `Matrix09.cpp` and `Vector09.cpp` from Week 9 (either your own version from the lab, or the version given with the Week 9 notes).

When you run this programme, it will prompt you for your ID number, and than create a `CSV` file that contains the entries for the adjacency matrix for your network, stored in CSV format. The graph is random, but seeded with your ID number, and so is unique to you. For example, the graph shown in Figure 1 is for a person whose ID number is 123. Their file is called `00000123.csv` and has the contents

```
0,0,0,0,1,0,1,0,0,0
0,0,0,0,0,0,0,0,1,0
1,0,0,1,0,0,0,0,0,1
0,0,0,0,1,0,1,0,0,0
1,0,0,0,0,1,0,1,0,0
0,0,0,0,0,0,0,0,1,0
0,1,0,0,0,0,0,0,0,0
0,1,0,0,0,1,0,0,0,0
0,0,0,0,0,0,0,1,0,0
0,0,1,1,1,0,0,1,0,0
```

Use the `MakeGraph` program to generate your own graph.

Write a C++ program that reads this file, and stores its contents in a matrix. From this matrix, generate the Markov matrix, S and the so-called "Google" matrix, G.

If you have time, form a vector $u = (1/N, 1/N, \ldots, 1/N)^T$, and compute Gu, G^2u, G^3u, ...

3 Visualising your graph

There is a graph description language called DOT. It is used by the GraphViz package, which can give nice visualisations of your graph.

In `http://www.maths.nuigalway.ie/~niall/CS319/lab7` folder you will find a file called Adj2Dot.cpp that reads an adjacency matrix from a "`CSV`" file, and generates the equivalent "`gv`" (GraphViz) file, can contains a description of the graph in Dot.

Usually, one installs GraphViz locally. But there are web versions available, such as `http://www.webgraphviz.com`, `http://sandbox.kidstrythisathome.com/erdos/`, and `http://graphs.grevian.org/graph`

Use `Adj2Dot` to generate the "dot" version of your graph. Then open that file in an editor, and copy its contents to `http://graphs.grevian.org/graph` to visualise your graph.

4 What next?

You don't have to submit anything from today's class. But we will build on it next week. In particular you will

1. Adapt the code to generate different types, and sizes of graphs;
2. Implement the PageRank algorithms;
3. Implement other algorithms that we discuss.