

In this lab you will perform some basic link analysis and calculate *PageRank* scores for a small section of the Web. The first stage is to represent a section of the web as an *adjacency matrix* and use it to find various linkage statistics. The second stage is to use this adjacency matrix to calculate *PageRank* scores for the pages. You do **not** need to perform a crawl of the internet for this labsheet although it does logically follow Labsheet 3. For convenience, older crawler data is being reused.

Part 1: Linkage statistics

1. Download the file:

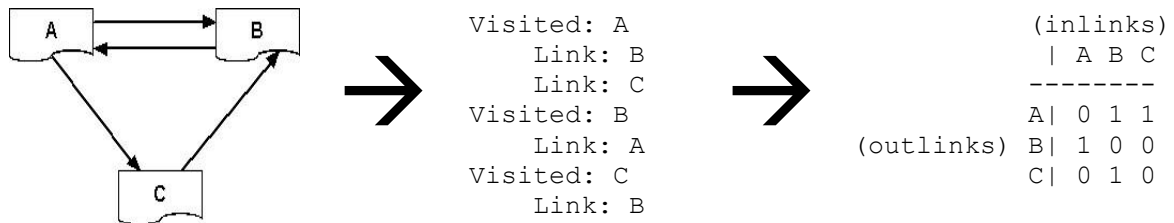
<http://www.dcs.bbk.ac.uk/~martin/sewn/ls4/sewn-crawl-2011.txt>.

This file contains data from a crawl of 500 pages. Open the file and you will see it has the format:

```

Visited: <URL of first page>
  Link: <URL of first link in page>
  Link: <URL of second link in page>
Visited: <URL of second page>
  Link: <URL of first link in page>
...
    
```

You are to write a program that will read this file and **extract the link graph** for this section of the web. You should represent the graph as an in-memory *adjacency matrix*. For a definition of an adjacency matrix see <http://mathworld.wolfram.com/AdjacencyMatrix.html>. For example:



As the crawl of the web is only partial (**all** crawls are – even for commercial search engines), you will need to make a decision about how to deal with pages that are listed as *Links*, but which are not *Visited*.

2. Once you have created the adjacency matrix use it to compute the following statistics (your program can output to the screen or to file(s), but you need to submit the results in a *ZIP* file):
 - a) The number of inlinks to each page
 - b) The number of outlinks from each page
 - c) The average (mean) number of inlinks to a page, and the standard deviation
 - d) The average (mean) number of outlinks from a page, and the standard deviation
 - e) The *degree distribution* of inlinks and outlinks, i.e. tables such as the following:

# outlinks	# pages
1	p
2	q
...	r

# inlinks	# pages
1	x
2	y
...	z

Part 2: Calculating PageRank

3. **Familiarise yourself** with the formula for *PageRank*, and how it is iteratively calculated. The formula is as follows:

$$PR(W) = \frac{T}{N} + (1-T) \left(\frac{PR(W_1)}{O(W_1)} + \frac{PR(W_2)}{O(W_2)} + \dots + \frac{PR(W_n)}{O(W_n)} \right)$$

Check that you follow the example *PageRank* calculation from Mark's lecture slides at: http://www.dcs.bbk.ac.uk/~mark/download/lec3_link_analysis.ppt. A useful article with further explanation can be found at <http://www.webworkshop.net/pagerank.html>.

SPEND ENOUGH TIME INITIALLY TO MAKE SURE YOU FULLY UNDERSTAND THE ALGORITHM BEFORE PROCEEDING TO WRITE CODE!

4. Next you will extend the program from 2 above to compute the *PageRank* for the pages listed in the crawl file.

To be able to calculate the *PageRank* scores, for each page you will need to know:

- the set of pages that link to it (the *backlinks* or *inlinks*).
- the set of pages that it links to (the *forward links* or *outlinks*).

This information can be easily extracted from the adjacency matrix you have generated – just read the ‘1’s along each row or column.

Your program should calculate and output the *PageRank* for each page in the site, using the teleportation constant $T = 0.15$, the value allegedly used in Google’s calculations.

You will need to decide on a suitable number of iterations to run the calculation in each case – **think carefully** about your criteria for convergence; do not end the calculation too soon.

The program output should be to a file called `pagerank015.txt`, of the form:

```
# iterations: <iter-Value>
              <URL1>         <PageRank1>
              <URL2>         <PageRank2>
              ...
```

Run the program again, this time with $T = 1.0$ and output to a file called `pagerank100.txt` - what do you notice about the time to convergence and the *PageRank* values in this case?

5. [*This final part is not compulsory for the lab, but if you complete it you can score ‘bonus’ marks*]

Run the *PageRank* calculation again using different teleportation values:

$T = 0.00$ $T = 0.50$ $T = 0.70$

Analyse your results (including those for $T=0.15$ and $T=1.00$):

- Do the results converge to the same value of *PageRank* no matter what the teleportation probability is?
- What difference does the teleportation probability make to the number of iterations necessary?
- Can you see any reason to choose 0.15 as a ‘standard’ value for teleportation?
- Is there anything else interesting you can see in the pattern of results?

What to hand in:

Send a *single ZIP* file by email to martin@dcs.bbk.ac.uk containing:

1. A file or files with all **the statistics from 2 above**.
2. **Your program code**, along with instructions for compiling and running it.
3. **Your program outputs** (i.e. the files named `pagerankXXX.txt`), showing the results of your *PageRank* calculations.
4. [*OPTIONAL*] **One A4 page** outlining your observations from 5 (MS Word or .pdf).

Submission Deadline: 01 Dec 2011.

Late assignments: No extensions are available as for this lab sheet and any late submissions will be graded as per the guidelines of the relevant MSc course being studied.

Miscellaneous:

Any emails sent to martin@dcs.bbk.ac.uk for this coursework should be headed SEWN Lab sheet 4.