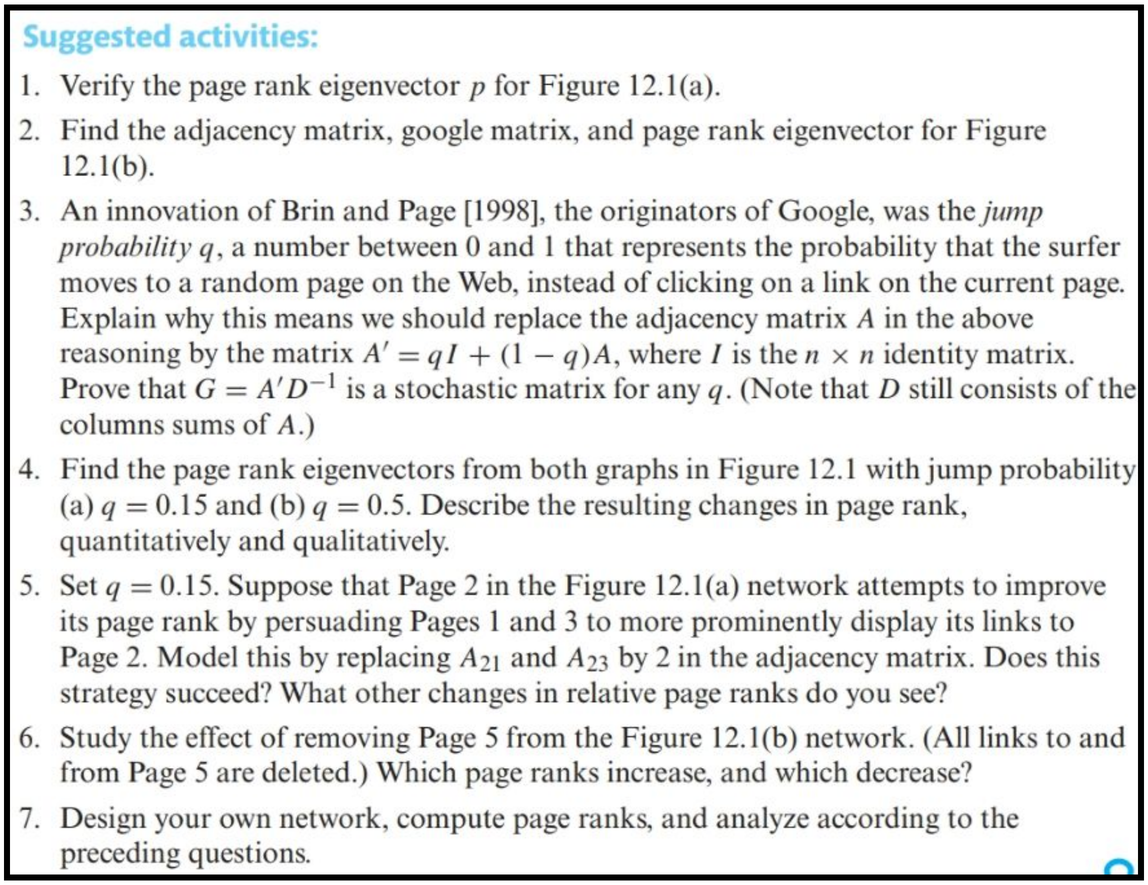
Numerical Analysis

Work for Matthew Fried

Chapter 12:



Suggested activities:

1. Verify the page rank eigenvector p for Figure 12.1(a).
2. Find the adjacency matrix, google matrix, and page rank eigenvector for Figure 12.1(b).
3. An innovation of Brin and Page [1998], the originators of Google, was the *jump probability* q, a number between 0 and 1 that represents the probability that the surfer moves to a random page on the Web, instead of clicking on a link on the current page. Explain why this means we should replace the adjacency matrix A in the above reasoning by the matrix $A' = qI + (1-q)A$, where I is the $n \times n$ identity matrix. Prove that $G = A'D^{-1}$ is a stochastic matrix for any q. (Note that D still consists of the columns sums of A.)
4. Find the page rank eigenvectors from both graphs in Figure 12.1 with jump probability (a) $q = 0.15$ and (b) $q = 0.5$. Describe the resulting changes in page rank, quantitatively and qualitatively.
5. Set $q = 0.15$. Suppose that Page 2 in the Figure 12.1(a) network attempts to improve its page rank by persuading Pages 1 and 3 to more prominently display its links to Page 2. Model this by replacing A_{21} and A_{23} by 2 in the adjacency matrix. Does this strategy succeed? What other changes in relative page ranks do you see?
6. Study the effect of removing Page 5 from the Figure 12.1(b) network. (All links to and from Page 5 are deleted.) Which page ranks increase, and which decrease?
7. Design your own network, compute page ranks, and analyze according to the preceding questions.

1.) I chose to use the power iteration method for several reasons. Firstly, we know the matrix is stochastic (proven in question 3) as such the largest eigenvalue is 1 and the power iteration method converges very quickly to the largest eigenvalue and eigenvector.
Secondly, the other main choice, the QR algorithm, has some downsides in this circumstance. We did not need to find the eigenvectors for every eigenvalue (just the largest) and, while the QR algorithm efficiently outputs the eigenvalues, due to the deflation step, it is more difficult than necessary to use it to find eigenvalues within the algorithm itself.
Furthermore, the QR algorithm is not efficient for large, sparse matrices. While superfast algorithms exist for structured matrices, such as Toeplitz and Hankel matrices (see anything by Victor Pan), one can often approach a sparse matrix by breaking it into blocks rather than just applying the general QR algorithm.
An interesting extension would be to look into “exotic” google matrices, that would effectively allow a larger rule set and thereby give a deeper look into web consumption habits and trends. I suspect that such a rule set could be generalized to the canonical form used when dealing with Markov Chains.

Code attached.

2.)

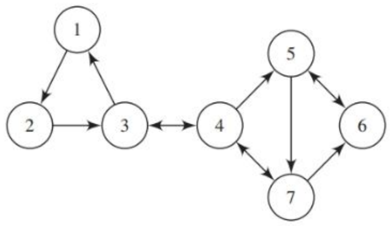
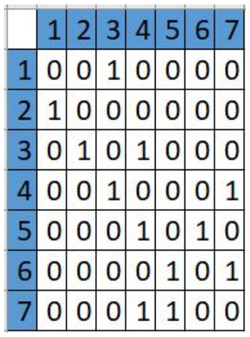


Fig. 12(b)



	1	2	3	4	5	6	7
1	0	0	1	0	0	0	0
2	1	0	0	0	0	0	0
3	0	1	0	1	0	0	0
4	0	0	1	0	0	0	1
5	0	0	0	1	0	1	0
6	0	0	0	0	1	0	1
7	0	0	0	1	1	0	0

Adjacency matrix for the figure. The rest is in the code.

3.) The A' matrix takes the probability q and places it in the diagonal of matrix A. We then scale A by (1-q). This convex function is simply scaling A and breaking apart its values into different paths (i -> j), it does not change the eigenvalues of the matrix. When we multiply by inv(D) we can think of it as forcing A into a stochastic presentation, by scaling the values across the columns.

4.) See attached code. What I found most interesting about both results is that changing q to higher values only seemed to change the probability of the largest value in the eigenvector. As such, the website with the largest traffic (i.e. highest probability) seemed to gain as people randomly surfed the web more often; and, conversely, weak websites lost “value” as eyeballs wandered off randomly.

5.) See attached code. Interestingly, page 2 gained tremendously, but all other websites except for the largest lost, not only page 1 and 3. It is surprising that the largest website gained when there was less spread among other sites.

6.) See attached code. The spread of the gains is extremely surprising. Page 2 and 3 effectively doubled, one did not change and 4 gained a little. It is very surprising how much a shift in the network topology affects the distribution. I think this could be an interesting area of research.

7.) Rather than design some specific network topology, I created a program that creates random matrices and runs the google page rank algorithm on those matrices. It stores the matrices in an array and it tests each eigenvector for every q from .05 to ,95 against a base case of q = 0 and then stores each value as a standard deviation from the base case. This can be used to see which matrices more or less affect the page rank.

Bibliography: I did not use any other sources for this project, but I will list those that I recommend and why.

1. Structured Matrices and Polynomials: Unified Superfast Algorithms, by Victor Y. Pan, ISBN-10: 1461266254 - the name should be self-explanatory