Mini-Project #3 — PageRank

Due November 17 at 11:55pm

1 Motivation

Please read the following papers on Google and PageRank:
http://www.sciencedirect.com.ezproxy.cul.columbia.edu/science/article/pii/S016975529800110X
http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf

2 Implementation

We're given a directed web graph (If we were to implement a full search engine, we would have a web crawler generate this graph), where each node represents a website and each edge a link, and we would like to know the order of importance of the websites in this graph.

Imagine a web surfer that obeys the following surfing pattern. dampingFactor is a fixed positive real number strictly smaller than 1:

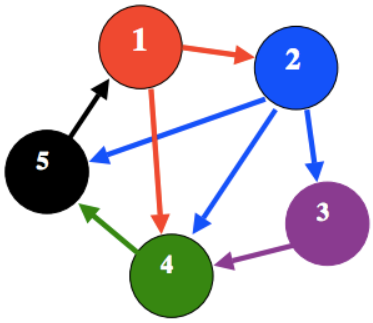
1. Go to a random page on the web graph.
2. If the page contains no links, go to step 1
3. Pick a random number between 0 and 1
   (a) If the number is greater than dampingFactor, go to step 1
   (b) Otherwise
      i. Click on a randomly chosen link on this page
      ii. Go to step 2

The PageRank of a website indicates the probability that this hypothetical web surfer is visiting the aforementioned site at any given moment. It provides a simple, yet powerful way to gauge the importance or popularity of a given website.

It is possible to calculate the PageRank of a set of websites in several ways. We will be using the power method, a popular and fast approximation method.

Let N be the number of nodes in our web graph. Let $\mathbf{Q}$ be an $N \times N$ matrix such that $\mathbf{Q}_{i,j}$ is the probability that the web surfer described above navigates immediately to page i after page j.

For example, for the following web graph



and a damping factor of 0.9, we have the matrix

$$\mathbf{Q} = dampingFactor \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 0 \\ 1/2 & 1/3 & 1 & 0 & 0 \\ 0 & 1/3 & 0 & 1 & 0 \end{bmatrix} + \frac{(1 - dampingFactor)}{5} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

If we have N websites on our web graph, we compute the PageRank vector p as follows:

$$\begin{aligned} \mathbf{p}^{(0)} &= \mathbf{Q} \cdot \mathbf{u} \\ \mathbf{p}^{(1)} &= \mathbf{Q} \cdot \mathbf{p}^{(0)} \\ &\cdots \\ \mathbf{p}^{(k)} &= \mathbf{Q} \cdot \mathbf{p}^{(k-1)} \end{aligned}$$

where u is a uniform probability vector of length N (a vector consisting of the number $1/N$ repeated N times).

The sequence $p^{(0)} \cdots p^{(k)}$ will convergence to the dominant eigenvector of matrix Q which is the expected solution of the pagerank problem.

Notice that, for testing purposes, p should be normalized, i.e. the sum of its elements should be equal to 1.

Your program should return an approximation of the vector $\mathbf{p}$ in the form of a Rail of Doubles and will be expected to be within *epsilon* distance of the correct solution $\mathbf{p}$.

3 Logistics

We will be testing your program on spicerack using two places with X10_NTHREADS set to 24 in each place.

The testbench will parse a test configuration from file and invoke your solver method.

The format of a configuration file is explained inside the test configuration file provided.

As usual include a WRITEME with your design choices. *Only submit solver.x10 and WRITEME.txt on Courseworks* inside a zip or tar file with the usual naming convention.