Project 3 - 25 points

In this project we will be working with large web graph data. In the following, you are expected to implement everything **from scratch** in the programming language of your choice. In other words, you cannot use libraries that perform the requested functionality (i.e., you have to really implement PageRank instead of using something from scikit-learn or downloading ready code from wikipedia or other sources, for example).

Please download the web-Google.txt.gz dataset with a **directed Web graph** from here: https://snap.stanford.edu/data/web-Google.html

In this project, we will be implementing PageRank. Although most of the details that you will need for the project are described below, you are encouraged to also go through the following papers to get a better understanding of the PageRank algorithm as well as some additional examples:

- The PageRank Citation Ranking: Bringing Order to the Web
- PageRank: Standing on the shoulders of giants

Please note that, in the description below and in the papers above, the examples are given in a matrix notation, i.e. there is an adjacency matrix used together with the PageRank vector. You are welcome to use matrices for this implementation but, if you find that you run out of memory, you may want to consider using a more sparse representation, i.e., a node pointing to a list of other nodes instead of using the full matrix.

PageRank (25 points)

PageRank captures the importance of every page on the Web. A page is important if it is pointed by many important pages.

Consider that we represent the Web with a Web graph matrix $M = \{ m_{ij} \}$. Each page i corresponds to row i and column i of the matrix M. In this matrix M:

- $m_{ij} = 1/n_j$ if page i is one of the n_j children of page j
- $m_{ij} = 0$ otherwise

In the above, pages can point to themselves and we would count the same page as a child of itself. For example, if page 1 points to pages 1 (itself), 10 and 20, then $m_{1,1}=1/3$, $m_{10,1}=1/3$ and $m_{20,1}=1/3$.

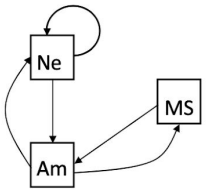
The simple PageRank equation that we want to compute is the following:

$$p = M \cdot p$$

where p is a vector of size Nx1 (where N is the number of nodes in the graph). p[i,1] represents the PageRank of page i. Initially every page's PageRank is equal to 1. We repeat the above computation, i.e., multiplying M with p and storing in p for a pre-specified number of steps. One high-level interpretation is the following: Initially every page has a unit of importance. At each round, each page shares its importance among its children and receives new importance from its parents. Eventually the importance of each page reaches a limit.

Here is a simple example of a PageRank computation for a few steps of a very small graph:

$$\begin{matrix} & p & & M & & p \\ \begin{bmatrix} n \\ m \\ a \end{bmatrix} & = & \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 \\ 1/2 & 1 & 0 \end{bmatrix} & & \begin{bmatrix} n \\ m \\ a \end{bmatrix} \end{matrix}$$



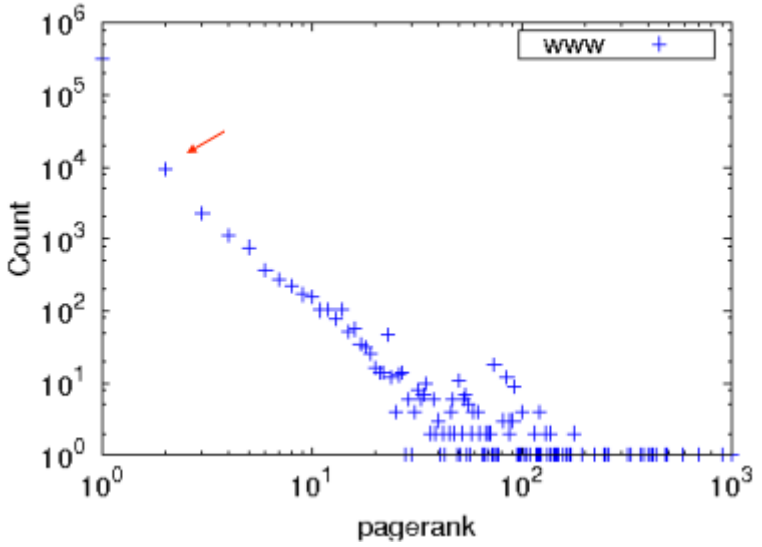
Iteration	0	1	2	3	...	
n	1	1	5/4	9/8	5/4	6/5
m	1	1/2	3/4	1/2	11/16	3/5
a	1	3/2	1	11/8	17/16	6/5

You are asked to:

a) (5 points) Implement the simple version of PageRank as was just described. Run your code for 10 iterations on the web-Google.txt.gz and provide the Web page ids with the top-20 and bottom-20 PageRank scores and also those scores.

b) (10 points) Implement the following improved version of PageRank: $p = \alpha \cdot M \cdot p + (1-\alpha) \cdot I_N$, where α is a numeric value, N is the number of pages (nodes) in the Web graph, and I_N is a unary vector (i.e., all values are set to 1) of size Nx1. Compute the PageRank again for 10 iterations for α=0.2 and α=0.85 and provide the top-20 and bottom-20 Web page ids together with their PageRank scores for each of the two α values (top-20/bottom-20 for α=0.2 and top-20/bottom-20 for α=0.85).

For each of the two computations above (i.e., one for α=0.2 and one for α=0.85) please also provide a histogram of PageRanks, i.e. a graph where the horizontal axis is a range of PageRank values and the vertical axis is the number of pages having a PageRank value within that range. Here is one example of what we are looking for (Note: the following graph is not the answer to this question, it is used here simply to show what the expected axes and plot would look like. Your actual graph from this project will be different from the image below):



In this graph the axes are both in log scale.The way to read this graph, for example, for the point pointed by the read arrow is that about 10,000 pages (10^4 in the y-axis) have PageRank between 1 (10^0 in the x-axis) and 1.258 ($10^{0.1}$ in the x-axis). You are allowed to use existing

plotting libraries/software **only** for generating the graph/graphics, but you need to compute the values to be plotted yourself. What are your observations from this graph?

c) (5 points) Repeat a) and b) for 50, 100 and 200 iterations providing the same results (i.e. top/bottom pages and histograms). How long does it take to run your code in each case?

d) (5 points) What is the point that you could stop the iterations, i.e., p has converged and doesn't change any further? How would you detect that in your code so you could stop the computations earlier?

Deliverables

- All files for your project need to be zipped in **one single zip file.** The file name should be in the following format: Lastname_Firstname_Project_X.zip (change X with the respective project number i.e. 1 for project 1, 2 for the second project and so on).
- Please email the zip file to the instructor by the deadline. **There are absolutely no exceptions and no extensions.**
- Inside the zip file, please include a) your source code b) any results that are part of the project c) a README text file that explains how to compile (if needed) and run your code and d) a short and concise report (preferably pdf - no more than 10 pages) describing your work and your approach to solving the project. **Please do not include the original or any derived data in the zip file.**

Important Notes

- Please check the class Web site: http://www.di.uoa.gr/~antoulas/m111 regularly for announcements and/or clarifications to the project. Announcements will show up there and will also be sent to the mailing list.
- You are free to make any assumptions you may need to along the way as long as you document them in your report.
- The project is meant to be worked on by the student submitting and only that student. In the event that there is a submission not worked on by the student, then the student fails the class.
- Although students are expected (and encouraged) to chat among them on potential solutions and approaches, sharing code and solutions is strictly not allowed. In the event that two or more students provide submissions that have common pieces, everyone involved (regardless of who is at fault or who copied from whom) fails the class.