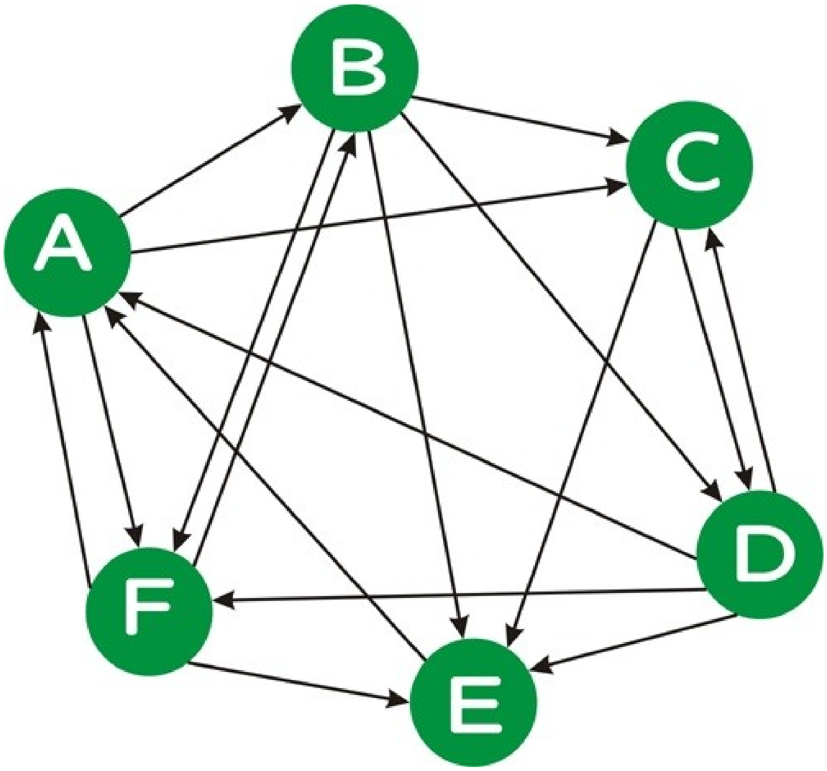
Programming Assignment 3

Q1) PageRank and Markov Chains

Consider the following directed graph:



a) Treat the above graph as a Markov chain, assuming a uniform distribution on the edges outgoing from each vertex. (In this problem part, you should *not* use any "teleportation.") Give the state transition matrix P of this Markov chain.

b) Compute the stationary distribution of this Markov Chain. This is a distribution π over the vertices such that

a. $\pi = \pi P$.

Note: In order to solve for π, you will need to solve six equations in six unknowns. Feel free to use a tool such as MatLab, if you like; otherwise, solve the equations by hand, eliminating one variable at a time. Also recall from class that the six equations given from $\pi = \pi P$ are not linearly independent; you will need to use five of these equations, together with the equation which specifies that the sum of the π probabilities must be 1.

c) Starting with the uniform distribution

a. $\pi^{(0)} = (1/6, 1/6, 1/6, 1/6, 1/6, 1/6)$

as an initial "guess", multiply $\pi^{(0)}$ by P to obtain a new "guess" $\pi^{(1)}$. Repeat this process, obtaining $\pi^{(n)}$ from $\pi^{(n\text{-}1)}$ via

$$\pi^{(n)} = \pi^{(n\text{-}1)}\ \mathrm{P}$$

until each of the π values are accurate within two decimal places (i.e, ± 0.01) of the values you solved for above. How many iterations are required?

Note: You should probably write a short program to do this, and this program will be useful for the problem part below as well.

d) Consider the PageRank formula as described in class and at the Wikipedia PageRank page. In particular, consider the following PageRank formula described on that page

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

(formula image courtesy Wikipedia). Let d = 0.85 be the damping factor.

Demonstrate that for the graph above, this formula is equivalent to computing the stationary distribution of a Markov chain described by transition matrix P', where each entry p'_{ij} in P' is obtained from the corresponding entry p_{ij} in P as follows:

$$p'_{ij} = (1\text{-}d)/N + d\ p_{ij}.$$

Using the matrix P' and your code from the problem part above, solve for the PageRank values of each vertex. (Start with a uniform distribution for $\pi^{(0)}$ and repeatedly multiply by P' until the π values "converge", e.g., they no longer change in the second decimal place). How do the PageRank values compare to the original stationary distribution values you computed above (and why)?

Submission: Submit answers to this question in handwritten form in class on 27 Oct 2017

Q2) In this assignment, you will compute PageRank on a collection of 183,811 web documents. Consider the version of PageRank described in class. PageRank can be computed iteratively as show in the following pseudocode:

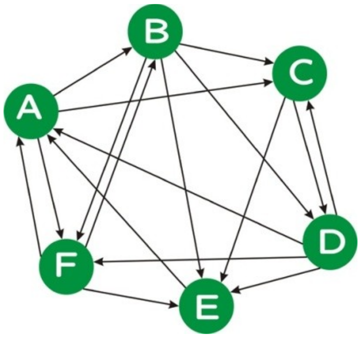
```
// P is the set of all pages; |P| = N
// S is the set of sink nodes, i.e., pages that have no out links
// M(p) is the set of pages that link to page p
// L(q) is the number of out-links from page q
// d is the PageRank damping/teleportation factor; use d = 0.85 as is typical

foreach page p in P
  PR(p) = 1/N                          /* initial value */

while PageRank has not converged do
  sinkPR = 0
  foreach page p in S                  /* calculate total sink PR */
    sinkPR += PR(p)
  foreach page p in P
    newPR(p) = (1-d)/N                 /* teleportation */
    newPR(p) += d*sinkPR/N             /* spread remaining sink PR evenly */
    foreach page q in M(p)             /* pages pointing to p */
      newPR(p) += d*PR(q)/L(q)         /* add share of PageRank from in-links
*/
  foreach page p
    PR(p) = newPR(p)

return PR
```

In order to facilitate the computation of PageRank using the above pseudo code, one would ideally have access to an in-link representation of the web graph, i.e., for each page p, a list of the pages q that link to p. For example, given the following graph, one such representation would be as follows



A D E F

B A F

C A B D

D B C

E B C D F

F A B D

where the first line indicates that page A is linked *from* pages D, E, and F, and so on. Note that unlike the example above, in a real web graph, not every page will have in-links, nor will every page have out-links.

Task 1

Implement the iterative PageRank algorithm as described above. Test your code on the example graph given above using the input representation given above. Be sure that your code handles pages that have no in-links or out-links properly. (You can test on a few such examples.)

What to Submit

List the PageRank values you obtain for each vertex after 1, 10, and 100 iterations of the PageRank algorithm.

Task 2

Download the in-links file (//sandata/xeon/Maryam Bashir/Information Retrieval/wt2g_inlinks) for the WT2g collection, a 2GB crawl of a subset of the web. This in-links file is in the format described above for Q1.

Run your iterative version of PageRank algorithm until your PageRank values "converge". To test for convergence, calculate the perplexity of the PageRank distribution, where perplexity is simply 2 raised to the entropy of the PageRank distribution, i.e., $2^{H(PR)}$. Perplexity is a measure of how "skewed" a distribution is --- the more "skewed" (i.e., less uniform) a distribution is, the lower its preplexity. Informally, you can think of perplexity as measuring the number of elements that have a "reasonably large" probability weight; technically, the perplexity of a distribution with entropy *h* is the number of elements *n* such that a uniform distribution over *n* elements would also have entropy *h*. (Hence, both distributions would be equally "unpredictable".) Run your iterative PageRank algorthm, outputting the perplexity of your PageRank distibution until the perplexity value no longer changes in the units position for at least *four* iterations. (The units position is the position just to the left of the decimal point.) For debugging purposes, here are the first five perplexity values that you should obtain (roughly):

183811, 79669.9, 86267.7, 72260.4, 75132.4

What to Submit

1. List the perplexity values you obtain in each round until convergence as described above.
2. Sort the collection of web pages by the PageRank values you obtain. List the document IDs of the top 10 pages as sorted by PageRank, together with their PageRank values.

3. Submit a copy of your code, which should hopefully be relatively short. (The pseudocode for the iterative PageRank algorithm is 10 lines long, as shown above.)

   Submission: Submit all files in zip format on Slate