**PageRank**

In this project, you will compute PageRank on a collection of 183,811 web documents. 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 \*/

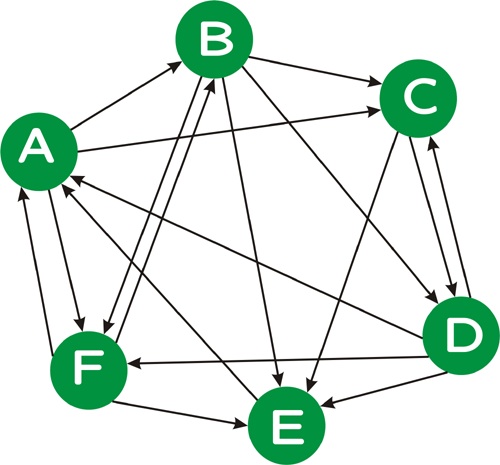
foreach page p

PR(p) = newPR(p)

return PR

In order to facilitate the computation of PageRank using the above pseudocode, one would ideally have access to an in-link respresentation of the web graph, i.e., for each page p, a list of the pages q that link to p.

Consider the following directed graph:



We can represent this graph 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 this example, in a real web graph, not every page will have in-links, nor will every page have out-links.

**The Project**

* Download the [in-links file](http://www.ccs.neu.edu/course/cs6200f13/wt2g_inlinks) for the [WT2g](http://ir.dcs.gla.ac.uk/test_collections/access_to_data.html) collection, a 2GB crawl of a subset of the web. This [in-links file](http://www.ccs.neu.edu/course/cs6200f13/wt2g_inlinks) is in the format described above, with the destination followed by a list of source documents.

Run your iterative version of PageRank algorithm until your PageRank values "converge". To test for convergence, calculate the [perplexity](http://en.wikipedia.org/wiki/Perplexity) of the PageRank distribution, where perplexity is simply 2 raised to the (Shannon) [entropy](http://en.wikipedia.org/wiki/Entropy_(Information_theory)) 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 perplexity. 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 algorithm, outputting the perplexity of your PageRank distribution 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, up to numerical instability):

183811, 79669.9, 86267.7, 72260.4, 75132.4

*To hand in:* List the perplexity values you obtain in each round until convergence as described above.

* Sort the collection of web pages by the PageRank values you obtain.