LAB 6: HITS + Page Rank + Trust Rank

1. **Motivation&Web Structure Mining:** Web structure mining is focused on discovering a topology of the web. In particular, a structure that is represented by hyperlinks (edges) that connect some pages (nodes). Links and web pages constitute a directed graph. The goal of web structure mining is to generate a structural summary of this graph. This can be, e.g.,:
    - finding similarities or relations between web sites,
    - categorizing web sites: for crawling policy, for ranking purposes (search engines),
    - revealing the structure of a web site (navigation purposes).

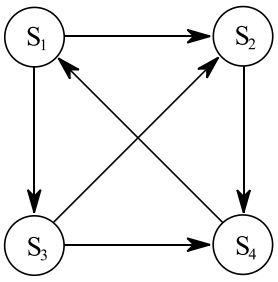
2. **HITS algorithm: HITS** (Hyperlink Induced Topic Search; John Kleinberg, 1998) – an algorithm for ranking Web Pages according to both their content and the structure of the links between pages in the Web. The algorithm divides pages into two groups:
    - authorities – pages to which many hubs link (contain important information),
    - hubs – pages that link to many authorities (show where to find an important information).

HITS is applied on a subgraph after a search is done on the complete graph. Firstly, the search is applied.Then,HITS analyses the structure of the links ofthe retrieved relevant pages.

Algorithm:

1. Identify the set of relevant pages by the standard text search (**a root set** R_q).
2. Extend the root set by adding:
    a. pages which are linked by the pages from R_q,
    b. pages that links to R_q.

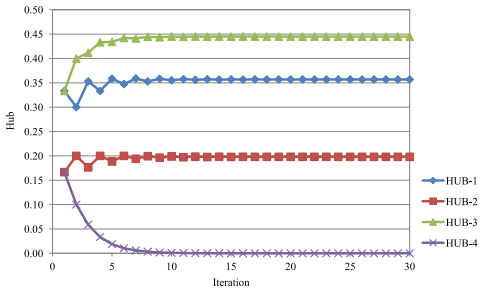
    The achieved (extended) set is called **a base set**S_q.
3. Then, method analyzes the structure of the base set to identify hubs and authorities:
    - Let L be the adjacency matrix of the S_q ($L_{i,j} = 1$, if page i links to page j, $L_{i,j} = 0$ otherwise).
    - Let $a = [a_1, a_2, \dots, a_n]$ be the vector of authorities and $h = [h_1, h_2, \dots, h_n]$ be the vector of hubs. These vectors contain coefficients (weights/importance). The greater the value of a_i (h_i)is, the better authority (hub) the i page is.h and a are defined as:
        - $h_i = \sum_{j:L_{i,j}=1} a_i$,
        - $a_i = \sum_{j:L_{i,j}=1} h_i$.
4. How to find the values of a and h?
    a. Iteratively:
        i. Initialize $a = [1, 1, \dots, 1]$ and $h = [1, 1, \dots, 1]$.
        ii. $h_i^{t+1} = \sum_{j:L_{i,j}=1} a_j^t$, $a_i^{t+1} = \sum_{j:L_{j,i}=1} h_j^t$
        iii. normalize a^{t+1} andh^{t+1} (sum of values in the vectors should equal 1).

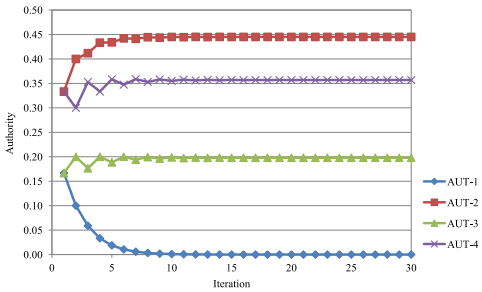


t = 0
h = [1, 1, 1, 1]
a = [1, 1, 1, 1]

t = 1
h = [2, 1, 2, 1]/6
a = [1, 2, 1, 2]/6

t = 2
h = [2/6 + 1/6, 2/6, 2/6 + 2/6, 1/6]/(10/6) = [3/10, 2/10, 4/10, 1/10]
a = [1/6, 2/6 + 2/6, 2/6, 1/6 + 2/6]/(10/6) = [1/10, 4/10, 2/10, 3/10]





b. Alternatively, using eigenvalues:

$$\begin{cases} h^{t+1} \leftarrow La^t \\ a^{t+1} \leftarrow L^T h^t \end{cases} \rightarrow \begin{cases} h^{t+1} \leftarrow LL^T h^t \\ a^{t+1} \leftarrow L^T L a^t \end{cases} \rightarrow \begin{cases} h^{\infty} = (1/\lambda_h) LL^T h^{\infty} \\ a^{\infty} = (1/\lambda_a) L^T L a^{\infty} \end{cases}$$

The last two equations are **eigenvalue equations** of LL^T and $L^T L$. One may compute h and a by finding eigenvectors that correspond to the greatest eigenvalue of LL^T and $L^T L$, respectively:

Compute hubs: LL^T
http://www.arndt-bruenner.de/mathe/scripts/engl_eigenwert2.htm
A. Find all λ by solving $\det(LL^T - I\lambda) = 0$. B. Select the greatest λ: [0.1980…, 1.000…., 1.554…, 3.246…]. C. Then, solve $(LL^T - I\lambda)x = 0$ for the selected λ. x is then an eigenvector. Solution $x = [0.801, 0.445, 1, 0]$. D. Normalize x by the sum of the elements of x to derive a vector of**hubs** h = [0.356, 0.198, 0.445, 0].
Compute authorities: $L^T L$
A. Find all λ by solving $\det(L^T L - I\lambda) = 0$. B. Select the greatest λ: [0.1980…, 1.000…., 1.554…, 3.246…]. C. Then, solve $(LL^T - I\lambda)x = 0$ for selected λ. x is then an eigenvector. Solution $x = [0, 1.246, 0.554, 1]$. D. Normalize x by the sum of the elements of x to derive a vector of**authorities** a = [0, 0.445, 0.198, 0.356].

c. Or by solving the system of equations. As you may observe above: $\lambda_h = \lambda_a$. Thus the following system of equations can be solved:

$$\begin{cases} h\mu = LL^T h \\ a\mu = L^T La \end{cases}$$

3. **PageRank algorithm: PageRank** (Page, Brin, 1998) – algorithm for ranking websites based on the structure of the links in the network. It assumes that the popularity of the page can be measured based on how often the average user visits a given page. PageRank uses "random web surfer" whichclicks links on pages with some probability and represents a random web exploration.

The idea of the algorithm is that the importance of the page is an average of the importance of the pages which links to it (d_i – page i):

$$PageRank(d_i) = v_i = \sum_{d_j : d_j \to d_i} \frac{v_j}{c_j},$$

where c_j is the number of links from the page d_j (including links to itself).

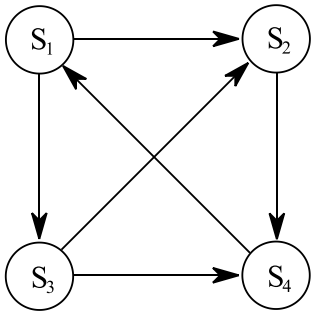
Applications:

- Search engines (websites ranking according to their PageRank; gives additional criterion based on the quality of a page, not only its content),
- predictions of Web traffic (estimation of users' visits count, server load, etc.),
- optimal crawling – crawlers should visit important pages more frequently (important page – a page with a high PageRank),
- website navigation,
- modeling ecosystems, protein networks.

Algorithm:

1. Construct stochastic matrix M:
   - M – matrix n x n, where n is the number of the pages,
   - $M_{i,j} = 0$, if the page j does not contain a link to the page i; $M_{i,j} = 1/c_j$, if the page j does contain a link to the page i.
2. How to find the values of PR?

   a. Iteratively**:**
      i. Let v be the vector of PRs, i.e., $v_i = PR(d_i)$. Set $v = [1/n, \dots, 1/n]$.
      ii. $v_i^{t+1} = \sum_j v_j^t M_{i,j}$,



t = 0

v = [1/4, 1/4, 1/4, 1/4]

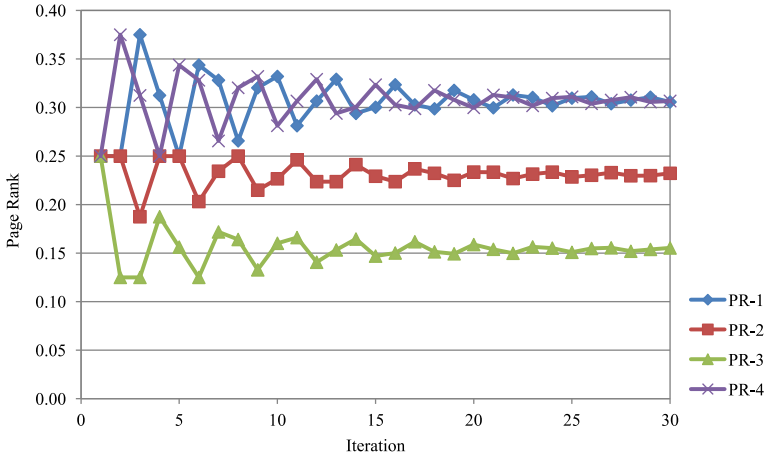
t = 1

h = [1/4, 1/4, 1/8, 3/8] / (8/8)

t = 2

h = [3/8, 3/16, 1/8, 5/16] / (16/16) = [6/16, 3/16, 2/16, 5/16]

$$M = \begin{matrix} 0.0, 0.0, 0.0, 1.0 \\ 0.5, 0.0, 0.5, 0.0 \\ 0.5, 0.0, 0.0, 0.0 \\ 0.0, 1.0, 0.5, 0.0 \end{matrix}$$



b. Alternatively, using eigenvalues: One may compute v by finding eigenvectors that correspond to the greatest eigenvalue of M:

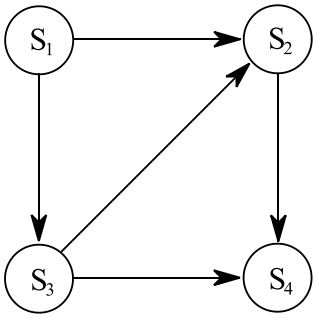
Compute v: M
http://www.arndt-bruenner.de/mathe/scripts/engl_eigenwert2.htm
A. Find all v by solving $\det(M - I\lambda) = 0$, B. Select the greatest λ: [-0.340…-0.8i, [-0.340…+0.8i, -0.319, 1 (!!!)] C. Then, solve $(LL^T - I\lambda)x = 0$ for the selected λ. x is then an eigenvector. Solution $x = [4, 3, 2, 4]$. D. Normalize x by the sum of the elements of x to derive a vector of **PRs** v = [0.307, 0.230, 0.153, 0.307].

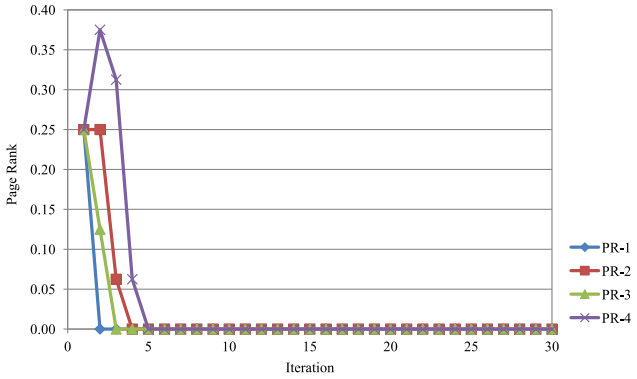
c. By solving the system of equations:

$$\begin{cases} v = Mv \\ \sum_i v_i = 1 \end{cases} \rightarrow \begin{cases} v_1 = 1v_4 \\ v_2 = 0.5v_1 + 0.5v_3 \\ v_3 = 0.5v_1 \\ v_4 = v_2 + 0.5v_3 \\ v_1 + v_2 + v_3 + v_4 = 1 \end{cases}$$

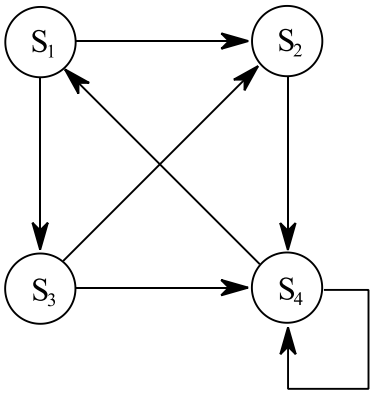
3. Page rank calculation problems ☹ :

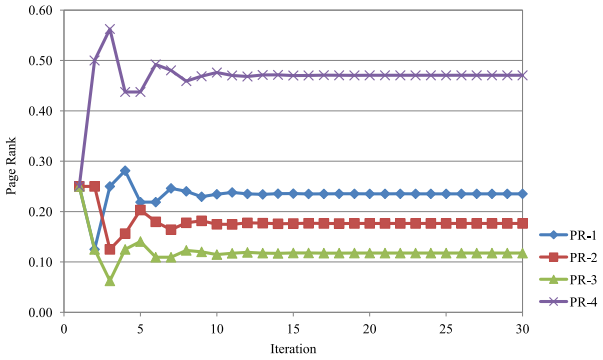
a. **Dead-end:**



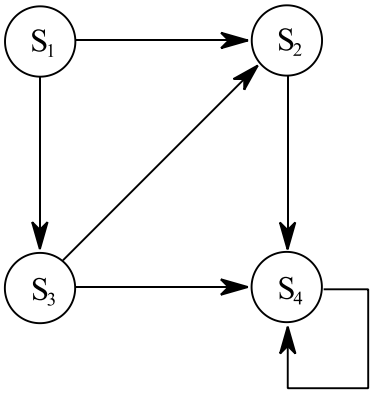


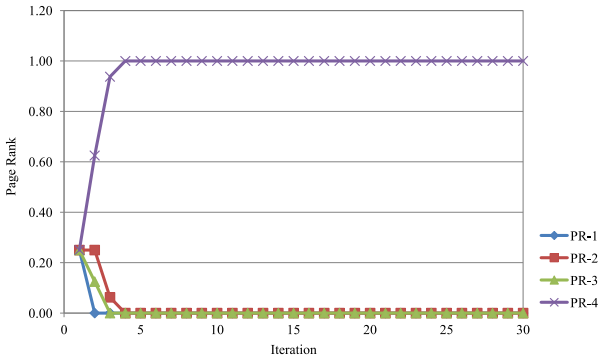
b. **Spider-trap:**





c. **Spider-trap + dead-end:**

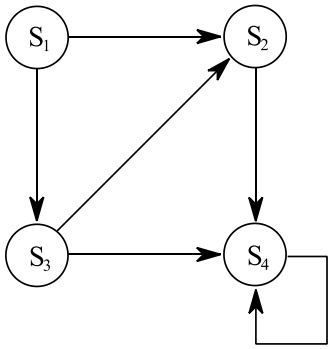


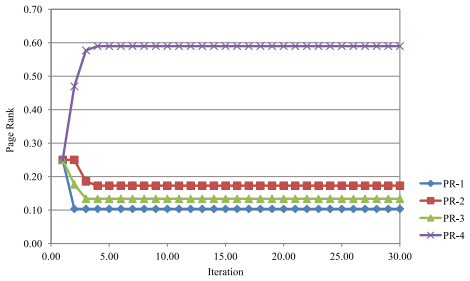


PageRank modification, which allow avoiding “spider traps” and “dead-ends”:

$$PageRank(d_i) = v_i = q + (1 - q)\sum_{\text{j}} \frac{v_j}{c_j},$$

where q is a *damping factor* (usually equals to 0.15). Example for q=0.15, normalization is applied so that the sum of PRs equals 1:



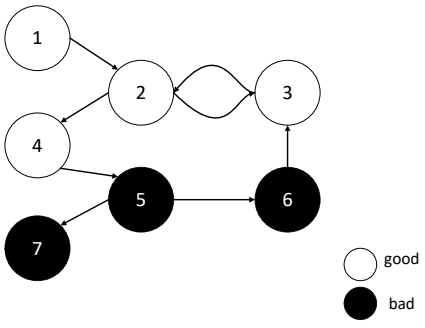


4. **Link farm:** a link spamming technique which goal is to maximize a page rank of some page. There exists three categories of web pages from the point of view of the spammer:

- pages that are not available – the spammer cannot add some content (comments, etc.),
- pages that are available – the spammer may add some content (comments, etc.),
- own pages.

Considering pages b) and c), the spammer may insert many links to some page **t** in order to increase a page rank of this page.

5. **<u>TrustRank algorithm</u>: TrustRank** is the modification of **PageRank** algorithm that main purpose is to handle spam (link spamming). The idea of this algorithm is that the situation, in which a “bad” page links go a “good” is very uncommon (approximate isolation):



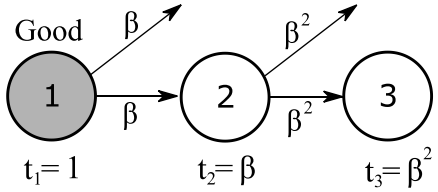
In **TrustRank** the trust factor is determined for each page.Then, based on some threshold, the pages are categorizedto: trusted and untrusted.

Algorithm

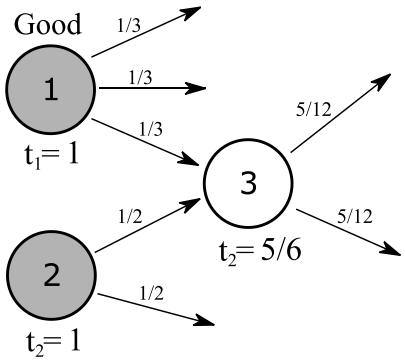
1. Choose a small subset of pages from the Web ("seed pages").
2. For each page from seed set determine whether it is "good" or "bad.
3. Good" pages are marked as "trusted".
4. Initially, the trust factor of pages is computed as follows:

$$O(p) = \begin{cases} 0, if\ a\ page\ is\ "bad" \\ 1, if\ a\ page\ is\ "good" \end{cases}$$

5. Propagate trust factor values according to the structure of the network:
   - Trust factor should decrease with the distance from the trusted page (distance – number of links between pages; **trust dampening**):



   - Trust factor should be divided equally for all pages linked from given page (**trust splitting)**:

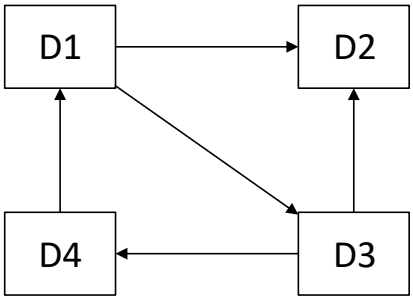


6. **TrustRank** as a modification to PageRank:
   a. Indicate a set of "good" pages and let $d_i = 1$ if the page i is trusted, $d_i = 0$ otherwise.
   b. Normalize $d \leftarrow d/\sum_i d_i$. Thus, $\sum_i d_i = 1$.
   c. Set an initial vector $TrustRank(d_i) = t_i = d_i$
   d. Apply the equation (e.g., iteratively):

$$TrustRank(d_i) = t_i = q * \boldsymbol{d} + (1 - q) \sum_{d_j : d_j \to d_i} \frac{t_j}{c_j},$$

LAB 6: Exercises

1. **HITS:** Given is the network shown in the image below. Find hubs and authorities vectors for this network. Complete the matrix L and L^T for this network and calculate matrix LL^T. Use online eigenvector calculator to find vectors h and a.



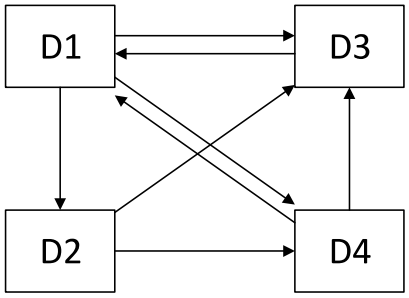
L = $\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$ L^T = $\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$

LL^T = $\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$ L^TL = $\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$

h = [, , ,], a = [, , ,]
h_{norm} = [, , ,], a_{norm} = [, , ,]

The best hub is page: ……., the best authority is page: …….

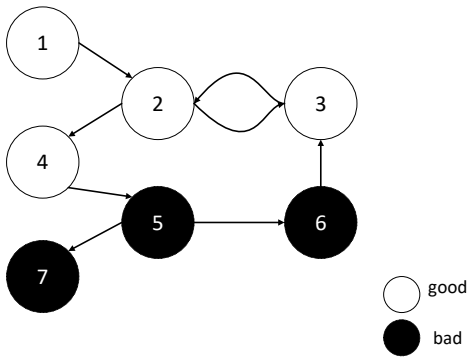
2. **PageRank:** Given is the network shown in the picture below. Find stochastic matrix M, write and solve the equation system for finding PageRank values for this network (use basic PageRank model – without a dumping factor).



M = $\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$

Equation system:

3. **TrustRank:** Find initial TrustRank vector d (seed = {2, 4, 5} and write equations for finding TrustRank for pages 2, 3, and 5, q = 0.15.



M =

d = [……., ……., …….,……..,……..,……..,……..]

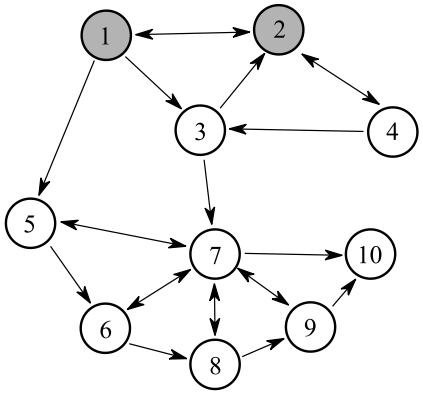
TR(2) =

TR(3) =

TR(5) =

4. **Programming Assignment (deadline +1 week)**

Given is the following web structure:



Download the pr_tr.py python script from the lab directory. The above structure is kept in L matrix (matrix of indices). Complete the TODOs:

TODO 1. Compute stochastic matrix M (function getM).

TODO 2. Compute pagerank vector and return the results (sorted pairs -> [page id : **pagerank**]). Which pages have the greatest pagerank? Why?

TODO 3. Which pages do you think belong to the link farm? Compute trustrank vector. Pages 1 and 2 are marked as “good”. Analyze the results. What has changed?

TODO 4. Repeat TODO3 but remove connections 1->5 and 3->7. Analyze the computed trustrank vector.