Lab session PageRank

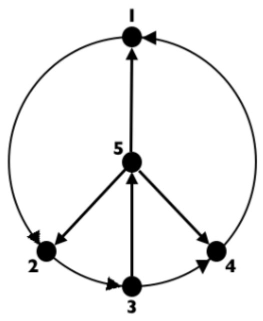
The lab session is done by team of two persons. The answers should be implemented with python, with report in format of Jupyter Notebook.
The objective of this lab session is to get students familiar with matrix manipulation with numpy, by implementing PageRank algorithm.
A link of Google Colab is recommended.

Exercise 1: PageRank

Compute the PageRank scores of the nodes in the graph G shown in Figure 1, without introducing random jumps. First, write down the system of linear equations and solve it. Is there a unique solution for the system of linear equations you wrote? Then, implement the PageRank algorithm and compute the PageRank scores with your code. Report the results so obtained. You can use any programming language for this task, you can also specify your own stop condition (e.g. the L1 norm between two PageRank vectors should be at most ε = 0.1). Does your code converge to the same vector, regardless of the initial scores?

Tips:

- use numpy for matrix representation. Here is a quick start tutorial https://numpy.org/doc/stable/user/quickstart.html
- You can use method *numpy.dot(A,B)* for inner product of two matrix A and B. (https://numpy.org/doc/stable/reference/generated/numpy.dot.html)
- You can use method *numpy.linalg.norm(x, ord=None, axis=None, keepdims=False)* to calculate the norm of matrix *x*. For example, if you want to calculate the L1 norm of matrix A, use *numpy.linalg.norm(A, ord=1)*



Exercise 2: PageRank with teleport

The file Hollins.dat contains 6012 URLs (from line 2 to line 6013) and 23875 linking relations (from line 6014 to line 29888). Apply PageRank algorithm you have written in Exercise 1 to this dataset. Give the top-k sites.

Tips: recommended procedures:

1. Load the file.
    a. You can create a dictionary to map the 6012 URLs and the id over line 2 to line 6013.
    b. You should create a stochastic matrix **M** of size 6012 x 6012, calculated from the link relation (see slide 8). (If you feel convenient, you can also create a matrix of link, then use numpy.count_nonzero() to calculate the value in M).
2. Implement PageRank with teleport to avoid spider trap (see slide 23)
3. For the dead ends, you can use a simple pruning strategy (see slide 26), i.e., remove all URLs without any successor. You should create a new dictionary as well as a new stochastic matrix for the pruned URLs.