Project 3: Page Rank implementations

Exercise 1: *Compute the PR vector M_m using the power method (adapted to PR computation). The algorithm reduces to iterate:*

$$x_{k+1} = (1-m)GDx_k + ez^t x_k$$

until $\|x_{k+1} - x_k\|_\infty < tol$

The corresponding code is attached on the file `C1.py`, which includes a function that uploads the binary matrix A, `uploadSparseMatrix(matrixName, n)`, providing the link matrix, a function that computes the multiplication of the sparse matrix by a vector, `multSparse(rows, cols, x0)`, and a function that conducts the page rank algorithm as described on the report, `pageRank(A, rows, cols)`. For more details on this algorithm consult *Appendx A*.
Note that in the `multSparse)` function, the multiplication is conducted taking into account that every nonzero coefficient will be a 1, therefore, the nonzero coefficients are indicating us which coefficients of the vector x_0 we should be adding up.

Exercise 2: *Compute the PR vector of Mm using the power method without storing matrices.*

The corresponding code is attached on the file `C2.py`, which includes a function that uploads the binary matrix A, `uploadSparseMatrix(matrixName, n)`, and a function conducting the page rank algorithm, `pageRank(A)`, this time without storing the matrix M_m and by considering each iteration step to be $x_{k+1} = M_m x_k$. In order to do so, the page rank algorithm follows the approach presented:

1. From the vectors that store the link matrix G obtain, for each $j = 1, ..., n$, the set of indices L_j corresponding to pages having a link with page j. This is done by noting that when expressing the sparse matrix in terms of the index array, `A.indices`, and pointer index array, `A.indptr`, the indices pointer array is an array in this case containing the sum of the number of indices corresponding to the first column, second column and so on. This is it indicates when we are skipping col. More formally, the pointer array has as coefficient j the number of elements until column j, i.e. $\sum_{i=1}^{j} n_i$.
   We want the array with the n_j indexes containing links, so for each column, this is for each k in the pointer index vector, we have n_k indexes in the index vector that correspond to the pages which have links with page k.

2. Each row of L_j has n_j elements.

3. Iterate $x_{k+1} = M_m x_k$ until $||x_{k+1} - x_k||_\infty < tol$.
   We do not want to compute explicitly the matrix M_m. Note that if $n_j = 0$, then we have $M_{ij} = 1/n$ for all $1 \leq i \leq n$. Then, recalling that $g_{ij} = 0$ iff $i \notin L_j$, then the product Mx can be implemented as follows[1]:

```
xc=x
x=0
for j in range (0,n):
    if(n[j]==0):
        x=x+xc[j]/n
    else:
        for i in L[j]:
            x[i]=x[i]+xc[j]/n[j]
x=(1-m)*x+m/n
```

[1]which is accordingly implemented using the logical notation from the rest of the project on the implemented code.

Appendix A

We already have uploaded the corresponding binary matrix: $A \in \mathbb{R}^{n \times n}$ where in this case $n = 36682$. Consider then the fixed point problem $Ax = x$. Remember that if the web network does not contain dangling nodes, then the matrix A is column stochastic, i.e. $1 \in Spec(A)$. If unique, the eigenvector of eigenvalue 1 is the so-called **PR vector**.
However for disconnected networks the PR vector is not unique. On the other hand, if the network has dangling nodes then the matrix A is column substochastic (and has no eigenvector of eigenvalue 1). In order to adress those two problems we may consider:

$$M_m = (1 - m)A + mS$$

where $0 \leq m \leq 1$ is a damping factor, which we may consider to be fixed at $m = 0.15$, and $mS = ez^t$, where $e = (1, \dots, 1)^T$ and $z = (z_1, \dots, z_n)^T$ is the vector given by $z_j = m/n$ if the column j of the matrix A contains non-zero elements (this is the node j has a link to any other node) and $z_j = 1/n$ otherwise.
Let $G = (g_{ij})$ be the link matrix, that is $g_{ij} = 1$ when there is a link between the pages i, j and $g_{ij} = 0$ if there is no link. Then, $n_j = \sum_i g_{ij}$ is the out-degree of the page j. Let now $D = diag(d_{11}, \dots, d_{nn})$, with $d_{jj} = 1/n_j$ if $n_j \neq 0$ and $d_{jj} = 0$ otherwise. Then $A = GD$
Now, if we seek to compute the PR vector of M_m using the power method, we may consider the algorithm given by the iteration of

$$x_{k+1} = (1 - m)GDx_k + ez^t x_k$$

until $\|x_{k+1} - x_k\| < tol$.