Exercise on Python and PageRank

Mauro Sozio
`name.lastname@telecom-paristech.fr`

In this lab session, we are going to learn and practice with Python and the PageRank algorithm. You should use IPython/Jupyter to edit and run your code. We recommend those who are not familiar with Python to check the tutorial on Python on the Web page of the course and proceed to Section 1. After that, move to Section 2. Those who are familiar with Python can go directly to Section 2. We recommend to use Python 3.0.

You should send us your Jupyter notebook including the answers to the questions and the code in Python. The answers for all lab sessions and the project should be sent together until the date specified on the website.

1 Practicing with Python

This section contains a few suggestions in order to practice with the basics of Python. This section will not be evaluated.

- construct a list of integers L. Then build another list M with the same size of L that contains the square of the elements in L.
- define a function f that receives in input a list of integers and returns a new list containing only even integers.
- write a few lines in Python that read a list of integers from a file and store them into a list L. Then run f on L.

2 Implementing PageRank in Python

The following questions give the same amount of points.

1. You should implement the PageRank algorithm in Python so as to deal with very large graphs. To this end, you should consider the algorithm in the last slide of the lecture on PageRank. The algorithm receives in input a directed graph G which is represented as a list of lines of the kind "i j" denoting that there is an edge between node i and j. The input is stored in a file, your code should receive the name of the input file as an argument. The output is the PageRank vector for G. For this question, we can assume that there are no dead ends in G. The matrix M_G should be represented using a sparse matrix representation, i.e. only non-zero entries should be represented. The product between the matrix M and the vector r should also be done while using the sparse representation

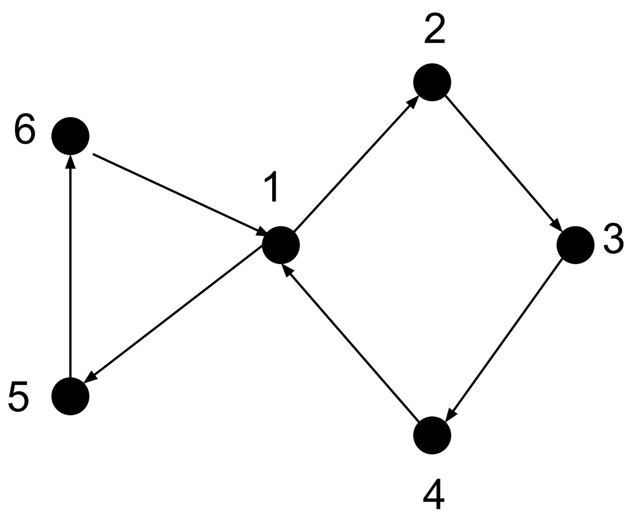

Figure 1: A simple web graph.

of M. Run your algorithm on the graph shown in Figure 2, while using first a) $\beta = 1$ and $\epsilon = 0.001$ and then b) $\beta = 0.8$ and $\epsilon = 0.001$. Report the results you obtain.

2. Construct the Web graph from the webpages extracted from Wikipedia provided to you ("WebPages.zip"). To do so, for each page, extract all links using the *findall()* method of the regular expression module *re* (`https://docs.python.org/2/library/re.html`) and add an edge from page i to page j if there is a link on page i to page j. Beware of duplicate link and selfloop. Tip: all links are preceded by 'a href="'. After having constructed the graph, run the PageRank algorithm on that.

Output: For each of the previous questions, your code should produce in output (e.g. using "print()") one or two lines containing the PageRank scores separated by a space character. E.g. "0.25 0.25 0.5" for a PageRank vector on three nodes. For question one you should produce two lines: the answer to question 1.a and the answer to question 1.b. It is important to follow these guidelines, failure to do so might result in the failure of the corresponding exercise.