Exercise on Python and PageRank

Mauro Sozio
Oana Balalau (TA), Maximilien Danisch(TA)
Jean Benoit Griesner(TA), Raphael Charbey(TA)
`name.lastname@telecom-paristech.fr`

In this lab session, we are going to learn and practice with Python. We suggest to use IPython to edit and run your code, however any other editors or IDE can be used. We recommend those who are not familiar 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.

1 Learning Python

This section contains a few suggestions in order to practice with the basics of Python:

- 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 Exercise on Python and PageRank

This exercises consists of implementing the PageRank algorithm in Python. It consists of three steps.

1. Implement the PageRank algorithm in Python. The algorithm receives in input a directed graph G which is represented as a list of lines of the kind "ij" denoting that there is an edge between node i and j. The output is the PageRank vector for G. In this step, 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. This allows to deal with very large graphs. The product between the matrix M and the vector r should also be done assuming M is sparse. For this exercise we can assume $\beta = 0.8$ and $\epsilon = 0.1$. For testing, the PageRank vector shown in Figure 2, is $\{\frac{2}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}\}$ with no random jumps ($\beta = 1$).

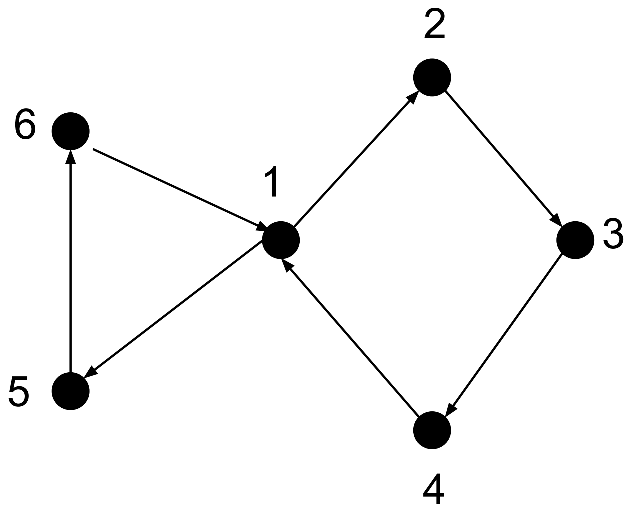

Figure 1: A simple web graph.

2. Implement in Python an algorithm that given an directed graph G in input, it removes dead-ends iteratively until no dead-ends are left.
3. Design a heuristic that: 1) removes the deadends from an input graph G, obtaining a graph H; 2) compute the PageRank vector for the nodes in H; 3) compute the pagerank vector for the nodes in $G \setminus H$ by starting from the PageRank scores of the nodes in H.
4. Run your code on directed and unweighted graphs downloaded from: `http://konect.uni-koblenz.de/networks/`. Try to scale your code to graphs as large as possible.