M2 – NSD (Practical Work 1)

Handling a graph

Maximilien Danisch, Marwan Ghanem, Lionel Tabourier

This first practical work aims at getting used to handling a large graph in main memory as well as gaining more insights on basic statistical notions of large interaction networks. In the following (and in future practical works), we represent networks by graphs. We denote a graph $G = (V, E)$, where V is the set of vertices (also called nodes) of the graph and numbered from 0 to $|V| - 1$ and E is the set of links (also called edges).
In the following (and in future sessions), *program* stands for a program or a combination of programs in the language of your choice.

1 To get things started

Exercise 1 — ***Preparation*** Create (manually) a few graphs and store them in files where each line is of the form:
x y
which indicates that a link exists between nodes x and y.
Download a few graphs of various sizes from:

- `http://konect.uni-koblenz.de/networks/` and
- `http://snap.stanford.edu/data/index.html`.

All these graphs will enable you to check the results of your programs.

Exercise 2 — ***Size of a graph*** Make a program that counts the number of nodes and edges in a graph (without storing it in memory) and writes this value on the standard output.

Exercise 3 — ***Node degree*** Make a program which counts the degree (*i.e.* the number of edges) of each node of a graph (without storing it in memory) and writes this value on the standard output.

Exercise 4 — ***Cleaning data*** Make a program that deletes the self-loops and duplicated edges existing in the graph (you can use unix commands to do this).

2 Load a graph in memory

Exercise 5 — ***Three graph datastructures*** Make three programs reading a graph and storing it in memory:

1. as a list of edges,
2. as an adjacency matrix,
3. as an adjacency list (if possible, the tables will be contiguous in terms of storage).

Note that these three programs are important as they will be used in the future practicals. Make sure to have them working fine.

Exercise 6 — ***Scalability*** Conclude on the scalability of the three programs.

3 A few basic network structural properties

Exercise 7 — ***Number of isolated nodes*** Create a program that computes the number of nodes of degree 0 of a graph, as well as the density, the average degree, the minimum and maximum degree of the graph.

Is it necessary to store the graph in memory for this purpose?

Let's remind that the *degree distribution* of a graph G is the function which associates to an integer k the number of nodes which have a degree k.

Exercise 8 — ***Degree distribution*** Create a program which computes the degree distribution of a graph and writes it in a file in the following format:
d n
where n is the number of nodes in the graph having degree d (you can use unix commands to do this).
Plot the obtained degree distributions (you can use GNUPLOT (see section 3)). What can you observe?

Mini-tutorial `gnuplot`

GNUPLOT is a program that allows to obtain a graphical representation of a structured dataset. There are a lot of online documentations and tutorials[1]. Here are only a few basic commands which will be useful to display your results.

- Launch and plot. You just need to execute the command `gnuplot` in a terminal and then `plot "file.txt" using i:j` in gnuplot command prompt to display the information contained in file *file.txt* using the values in column i (abscissa) and column j (ordinates).

- Ranges. If the minimal and maximal values on the axes are not appropriate:
  `plot [x1:x2][y1:y2] "file.txt"`
  allows to plot with x-axis values between $x1$, $x2$ and y-axis values between $y1$, $y2$.

  Another option is the command:
  `set xrange` $[x1 : x2]$: display data corresponding to values of x between $x1$ and $x2$ (same goes with `yrange`)

- Scales and labels. Turn to a logarithmic scale on both axes: `set logscale xy`
  Turn to a linear scale on both axes: `unset logscale xy`
  Give a specific label to an axis: `set xlabel` *name* : give the name *name* to x axis (same goes with `ylabel`).

- Saving. To save a picture in a file, you must first choose the kind of file with the command `set terminal`, for example for a postscript file:
  `set terminal postscript`
  Then you can choose a file name:
  `set output "my_plot.ps"`
  so that the plot is created in the corresponding file (in the current directory). Then use the corresponding `plot` command to the curve that you want to plot.

  Other format are available, for example: png, pdf, ...

[1]`http://www.gnuplot.info/`
`http://www.info.univ-angers.fr/~gh/tuteurs/tutgnuplot.htm`

Remark: notice that after an output file has been declared with `set output`, all the following commands will write in this output (so that the first one may be overwritten). To avoid that, create a new output, or if you want to print on screen, use:

```
set output "/dev/null"
set term x11
```