CPA. Handling a large graph

Maximilien Danisch, Clémence Magnien, Lionel Tabourier

1 To get things started

Exercise 1 — *Preparation* Download the following graphs:

- `http://snap.stanford.edu/data/email-Eu-core.html`
- `http://snap.stanford.edu/data/com-Amazon.html`
- `http://snap.stanford.edu/data/com-LiveJournal.html`
- `http://snap.stanford.edu/data/com-Orkut.html`
- `http://snap.stanford.edu/data/com-Friendster.html`

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

You can, in addition, create (manually) a few small graphs and store them in files where each line is of the form:
u v
which indicates that a link exists between nodes u and v.

Exercise 2 — *Size of a graph* Make a program that counts the number of nodes and edges in a graph and writes this value on the standard output.

Exercise 3 — *Cleaning data* We assume the graphs to be simple and undirected. Make a program that deletes self-loops and duplicated edges (e.g. for a bidirected edge (a line "u v" and then a line "v u" in the file), just keep one of them).
Note that you can use unix commands to do that and write the result in a new ".txt" file.

2 Load a graph in memory

Exercise 4 — *Three graph datastructures* Make three programs to read a graph and store it in memory:

1. as a list of edges,
2. as an adjacency matrix,
3. as an adjacency array.

Note that these three programs are important as they will be used in the future practicals. Make sure to have them working fine.
Use them on the 5 downloaded graphs and conclude on the scalability of the three programs.

3 Breadth-first search and diameter

Exercise 5 — *BFS* Implement an efficient BFS algorithm.

Use it to make an algorithm that outputs all connected components and their sizes (number of nodes).

Test your algorithm on the 5 downloaded graphs and, for each one of them, report the fraction of nodes in the largest connected component.

Use your BFS algorithm to make an algorithm that computes a good lower bound to the diameter of a graph.

Test your algorithm on the 5 downloaded graphs and report your lower bound as well as the running time of your algorithm.

4 Listing triangles

Exercise 6 — *Triangles* Implement an efficient algorithm for listing triangles.

Test your algorithm on the 5 downloaded graphs. For each graph, report the number of triangles as well as the running time of your algorithm.

Generalize your algorithm to compute the transitivity ratio[1] of the graph.

Test your algorithm on the 5 downloaded graphs and report the transitivity ratio of each graph.

Generalize your algorithm to count the number of triangles each node belongs to and to compute the clustering coefficient of each node[2] in the graph and the clustering coefficient of the graph[3]. Test your algorithm on the 5 downloaded graphs and report the clustering coefficient of each graph.

[1]The transitivity ratio is defined as three times the number of triangles devided by the number of V-edges (a V-edge is a pair of edges that share an end node).

[2]The clustering coefficient of a node is defined only if the node has 2 neighbors or more. It isthe number of triangles the node belongs to devided by its number of pairs of neighbors.

[3]The clustering coefficient of a graph is defined as the average clustering coefficient of all its nodes with two neighbors or more.