Storing and processing graph data

The first lecture provided a mathematical definition of a simple graph as an ordered pair of two sets, namely a set of nodes and a set of unordered/ordered pairs (for undirected/directed networks). It is clear that both sets must be included in the data representation. However, if nodes are sequentially numbered or indexed, storing the list of nodes is often obsolete. Then only the edges or the set of pairs of nodes has to be recorded.

A good starting point is the wikipedia page on ***abstract graph data types:*** https://en.wikipedia.org/wiki/Graph_(abstract_data_type) and, more formally, Butts' 2008 paper (see course documents) which gives a general introduction on network data, the representation, network boundaries and visualizations.

The wikipedia page lists three different representations:

1. Adjacency List (https://en.wikipedia.org/wiki/Adjacency_list)
2. Adjacency Matrix (https://en.wikipedia.org/wiki/Adjacency_matrix)
3. Incidence Matrix (https://en.wikipedia.org/wiki/Incidence_matrix)

The first two representations are most often used.

This page lists also a set of possible graph operations with their associated time complexity cost in each of the representations. Since our focus on network analysis in a big data setting, the Incidence Matrix representation is least preferrable.

Graph File Formats

Besides these elementary network representations, several data formats have been developed to store network data together with additional information. Each of these formats can be scored on the following properties

- Actual network representation as Matrix or List
- XML-format or plain text format
- Supports distinction between undirected and directed graph
- Supports distinction between unweighted and weighted graph
- Supports node labels
- Supports edge labels
- Supports multiple edges between two nodes
- Includes layout or visualization information
- Supports dynamic graphs

Several data formats have been developed for storing network data. Some of them are developed independently from graph processing software, others are specific to certain applications. Most software, however, can deal with many formats.

Graph Processing Libraries

The data formats are for storing and sharing network data. Another issue is the internal graph representation in graph processing software or libraries. I list a few open source Java libraries.

- JUNG, (Java Universal Network/Graph Framework, http://jung.sourceforge.net)
- JGraphX https://github.com/jgraph/jgraphx (Strong graphical focus)
- JGraphT (http://jgrapht.org)
- GraphStore (https://github.com/gephi/graphstore)

Task 1:

Investigate the source code of an open soure graph processing-library
How is the set of nodes and the list of edges implemented? Use the discussion forum for your findings.

Visualization

The two most used approaches for plotting networks are MDS-based techniques versus spring-models. A recent third approach is gaining popularity.

The Multidimensional Scaling approaches try to map the dissimilarities (or distances between nodes) in a two dimensional space. (See: https://en.wikipedia.org/wiki/Multidimensional_scaling for a brief introduction) while preserving these dissimilarities among pairs of nodes. The minimization of the cost function is calculated at the global level of the graph.
See http://www.mathpsy.uni-tuebingen.de/wickelmaier/pubs/Wickelmaier2003SQRU.pdf for a more formal introduction.

Spring based or force-directed models incorporate some attraction between linked nodes and possibly some repulsion between unlinked ones. These models try to maximize attraction and minimize repulsion.
http://cs.brown.edu/people/rtamassi/gdhandbook/chapters/force-directed.pdf

A third approach that builds of the original idea of Multi dimensional scaling as a dimensionality reduction approach is t-SNE (https://lvdmaaten.github.io/tsne/). This technique applies a nonlinear reduction of the high dimensional space into a two- or three-dimensional space preserving the probability that two similar points are nearby while dissimilar points are more distant.

Task 2:

Investigate existing graph processing software and make an inventory of implemented graph visualization algorithms. Discuss these techniques and try to identify the criteria that were used by the authors to obtain a 'good' visualization. Evaluate them with respect to Scalability.