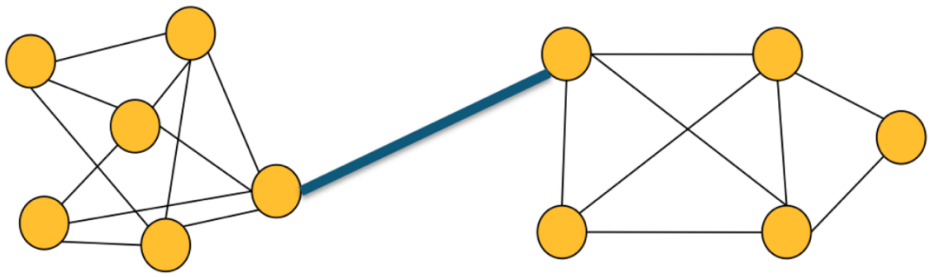
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Big Data Algorithms

TP1: Community Detection

A **community** is a set of nodes between which the interactions are (relatively) frequent: a **cohesive group of nodes** that are connected **"more densely"** to each other than to the nodes in other communities. Being able to identify these sub-structures within a network can provide insight into how network function and topology affect each other. One algorithm for finding communities was proposed by Girvan-Newman and relies on the notion of edge traffic.

The **traffic** of an edge is the **number of shortest-paths** in the network that **pass through that edge**. An edge with **high traffic** is a potential deal maker and is in a special position since most other nodes have to channel their communications through it:



The Girvan-Newmann algorithm uses the idea that “bridges” between communities must have high traffic. So, in order to detect communities, the idea is to find these edges with high traffic and to progressively remove them from the original network. The connected components of the remaining network are the communities.

To test the following exercises you can use the example at the end of CM2 (Appendix) from the course website: http://bit.do/algobd

Exercise 1. Edge traffic

Write a Java program containing a class that computes the traffic values for all edges in a graph (stored in a .txt file).

The traffic can be seen as a measure of the total amount of **"flow"** an edge carries between all pairs of nodes. A single unit of flow between two nodes divides itself evenly among all shortest paths between the nodes (1/k units flow along each of k shortest paths). For more details, on the traffic computation see the example in the course (from slide "Computing Edge Traffic").

For each node N in the graph:

1. Perform **breadth-first search** of graph starting at node N
2. Determine the **number of shortest paths** from N to every other node
3. Based on these numbers, determine the amount of flow from N to all other nodes that use each edge

Divide sum of flow of all edges by 2

Exercise 2. The Girvan-Newmann algorithm

Write a Java program that implements the community finding algorithm, using the previous class for traffic computation.

1. Compute traffic for all edges in the network
2. Remove the edge with highest traffic
3. Go to step 1 until no edges left

As you may have noticed, executing the whole algorithm leaves you with communities made up of single nodes. So, one important decision is **when to stop the iteration**. This is typically done by evaluating a **quality function** – which measures the "quality" of the partitions of a network into communities. There are different proposals for such functions, but they are out of our scope here – so feel free to stop the algorithm after 2-3 iterations.