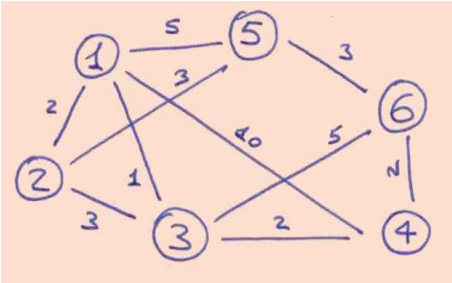
MEEF **Univ. Rennes 1**

Routing Tables and Flooding

The main aim of this assignment is to explore the different possible ways for a message, sent by a computer device belonging to a particular network, to find its best route among the other computers of the network. We won't be talking too much about *computers* and *networks*, but we will rather focus on the *graph* representation of the network.

All the exercises below will be based on the following graph G (the curious student may want to try, subsequently, to apply them to other graphs).



Routing tables

Routing tables are used in computer networks to keep information about the shortest paths that can be identified in the network. On every graph vertex, they do not contain the information about the *entire* path towards a destination vertex, but rather they store information about the next vertex to be visited in order to reach a given destination, together with the total distance to the destination vertex.

- Compute the routing table for the vertex 1 of our graph G; you are free to use any algorithm to this purpose.
- By exploring the result obtained above, we may wonder whether it is actually necessary to have a dedicated record in such tables for every distinct destination vertex. What is your opinion?

Flooding on graphs

The routing tables allow us to define the shortest path for a message to travel through the vertices of a graph so that every vertex can quickly and efficiently communicate with another. This approach allows us to compute the shortest paths only once, and to keep this information in the routing tables as far as the graph does not change its structure.

However, this approach does not take into consideration that the network is a dynamical entity: one of its vertices may be overcharged, or even currently not functional. Messages passing through these vertices are likely to be delayed ...

Flooding is therefore an alternative approach to finding paths on computer networks, where the best route is identified dynamically, on the basis of the current status of the network.

- Suppose that vertex 1 needs to send a message to vertex 6 over our graph G. To this aim, the vertex 1 sends this message to all its neighboring vertices; such neighboring vertices

will then be in change to send the message to their own neighboring vertices, until the destination vertex is reached. If the sent message contains the information about the sending and destination vertex (that the other vertices will exploit to decide whether to forward the message or not), is it actually possible to count the number of messages that are sent out in the graph?

- In order to reduce the total number of unnecessary messages that are sent over the graph, we decide to introduce another information in the messages: the *hop limit*, which establishes the maximum number of times that the message can be sent forward by the receiving vertices. Compute the total number of messages with hop limit equal to 2 and 1. What can you remark?
- Let's verify in more details what happens in vertex 3. Are there chances that vertex 3 forwards more than once the same message to another vertex? If yes, how to avoid this problem?