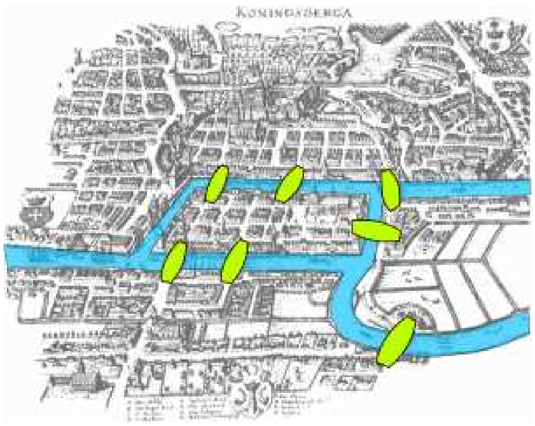
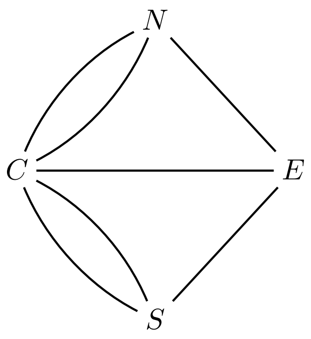
Information and Communication Technologies — S7 — Work n°12

Graphs — Part 1

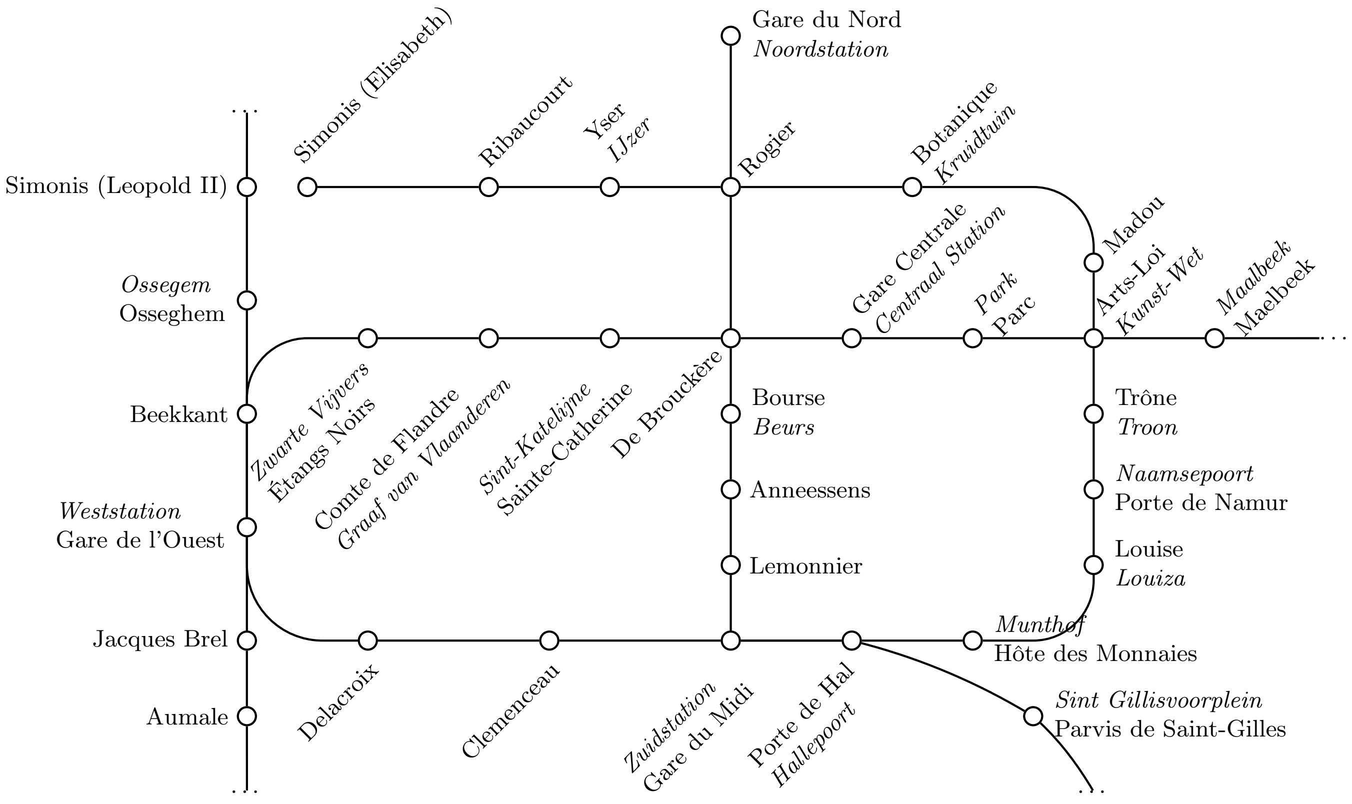
In 1736, Euler investigates the problem of the “Seven Bridges of Königsberg” : is there a path that goes on each bridge, and only once, in the city? There is none. Later, another question is to minimize the distance undergone by a postman that wants to go through every mailbox in the city, the “Travelling salesman problem”. To solve those problem, we must model the real world: the people to visit are modelled by vertices, the roads by edges ; and everybody together forms a graph. Königsberg can be represented by the graph on the righ, where S is the south side of the Pregolya river, N the north side, C the central island and E the eastern island:





Source : Wikimedia Commons.

If you have already seen a subway map, you have already seen a graph! Graphs are a way to represent networks of public transport[1]: vertices are the subway stops, and arcs indicate possible connections between two stops. For instance, this is an excerpt from the Brussels subway map:



[1] http://www.le-cartographe.net/blog/archives/107-la-representation-cartographique-du-metro

1 Definition

A simple oriented graph is given by a set V (the vertices) and a set $E \subset V \times V$ (the edges).

Sometimes, it will be convenient to note n the size of V (the number of vertices) and m the size of E (the number of edges).

Remark: an equivalent definition for the edges is to see E not as a set included in $V \times V$, but as a relationship $\mathcal{R}$ on V: in any case, we just need to describe which vertices are connected to which ones.

To given an "abstract" example to illustrate the definition, the couple ($V = \{a; b; c\}$; $E = \{(a, a); (a, b); (b, a); (b, c)\}$) is a graph containing 3 vertices and 4 edges. To better understand this notion of graph, the easiest thing to do is to draw it. To do so, we just write the different names of the vertices, and draw arrows from a vertex to another if the corresponding edge exists:

$$\begin{tikzcd}
a \arrow[loop left] \arrow[r, bend left] & b \arrow[l, bend left] \arrow[r] & c
\end{tikzcd}$$

Remarks:

- if i and j are two vertices, the existence of the edge (i, j) does not imply the existence of the edge (j, i). Said in another way, one edge from i to j does not guarantee the edge in the other direction (from j to i).
- we will cover only simple graphs, where there can be at most one edge from a vertex to another.[2]

2 Representing a graph

We have met the easiest way to represent a graph: the drawing. We'll cover two others.

2.1 Outgoing neighborhood

We will list, for each vertex i, the set of vertices j towards which there exist an edge starting from i, which is $\Gamma^{+}(i) = \{j \in V | (i, j) \in E\}$ (the outgoing neighborhood of i). On the drawing, it is the set of vertices towards which there is an arrow leaving i.

In the example, the outgoing neighborhood is:

$$\begin{tikzcd}
a \arrow[loop left] \arrow[r, bend left] & b \arrow[l, bend left] \arrow[r] & c
\end{tikzcd}$$

Vertex	a	b	c
Out-neighbors	a, b	a, c	$-$

2.2 Incoming neighborhood

We will list, for each vertex i, the set of vertices j from which there exist an edge ending at i, which is $\Gamma^{-}(i) = \{j \in V | (j, i) \in E\}$ (the incoming neighborhood of i). On the drawing, it is the set of vertices from which there is an arrow coming towards i.

In the example, the incoming neighborhood is:

Vertex	a	b	c
In-neighbors	a, b	a	b

2.3 Adjacency matrix

To construct this matrix, we need a "natural" order on the vertices (lexicographical order, naturao order on numbers...), and each vertex name will be replaced by its integer in the chosen order. The adjacency matrix M, a 2d-array of size $n \times n$, is defined by:

$$(M)_{i,j} = \begin{cases} 1 & \text{when the edge } (i, j) \text{ is in the graph} \\ 0 & \text{if it's not} \end{cases}$$

[2]When we can have more than one edge, it is a multigraph.

In the previous example, the adjacency matrix is:

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

When looking at the definition of the coefficient inside this 2d-array, we can see that:

- to read the outgoing neighborhood of a vertex i, we look at the 1s in the i-th line of M
- to read the incoming neighborhood of a vertex j, we look at the 1s in the j-th column of M

The adjacency matrix gives many useful computations on graphs, but we'll first stick to the neighbor lists.

3 With Python

Let's define a new class "Node", and define the graph in the previous example by using the outgoing neighborhoods. See Listing 1.

```python
class Node:
    def __init__(self, name):
        self.data = name
        self.out_neighbors = []

a = Node("a")
b = Node("b")
c = Node("c")

# We must first define the nodes, before we can define the neighbors (which use
    the nodes!)
a.out_neighbors = [a, b]
b.out_neighbors = [a, c]
c.out_neighbors = []

# Define the graph g as an array containing the three vertices a, b, c.
g = [a, b, c]
```

Listing 1: The Graph data structure.

Let's try to take a walk (on the wild side) on graphs, see Listing 2.

```python
from random import *
def random_walk(graph, nb_steps):
    if g == []:
        print("I can't move on an empty graph.")
        return
    index = randint(0,len(g)-1)
    current_vertex = g[index]
    print("I start from " + current_vertex.name + ".")
    for i in range(nb_steps):
        if current_vertex.out_neighbors == []:
            print("I can't move anymore.")
            break
        index = randint(0,len(current_vertex.out_neighbors)-1)
        current_vertex = current_vertex.out_neighbors[index]
        print("I move to " + current_vertex.name + ".")
```

Listing 2: Random walk on a graph.

1. Execute the code `random_walk(g, 50)` and explain what it does.
2. Design a graph with 5 vertices where the random walk will never stop, no matter what number of steps we make. Draw this graph on paper, create it in Python, and test the random walk.

3. In a given graph, a path from a vertex i to a vertex j is a sequence of edges where the end of one edge is the start of the next one. For instance, there is a path $a \to b \to a \to a \to b \to c$ from a to c, which has length 5 (it is made of 5 edges — an edge may appear multiple times in a path, like it is the case here).

   (a) in the example graph, count the number of paths of length 1 from a to a, from a to b, from a to c, from b to c, from c to c;

   (b) do the same for the paths of length 2;

   (c) and also for the paths of length 3.

   Remark: for those who already know how to manipulate matrices, the matrix M^p gives the number of paths of length p from each vertex to each vertex, where M is the adjacency matrix.

4. For each vertex in this example, create the set of the accessible vertices (the vertices that you can reach, by using any number of edges).

5. Draw the graph of the S7ICTE group. This graph contains 9 vertices (each vertex is a different student from the group), and there is an edge connecting two students if they have the same mother tongue (edges will not be directed this time, because if there is an edge in one direction, the edge in the other direction should also exist: therefore we can just draw undirected edges). This graph is disconnected. What can we say of the different parts of this graph?