Machine Learning on Graphs
Practical lab - session 2

S.Nicolas

September, 2025

Goal : The aim of this second practical lab session is to study the main algorithms which can be applied on graphs for graph traversal, shortest paths finding, or minimum spanning tree computation, by using the NetworkX library.

Instructions : during the session, you will perform the compulsory work requested below. You should preferably use a Python notebook for your implementation. You must submit your work at the end of the session on the UniversiTICE deposit space provided for this purpose. The optional work proposed in the "additional work" section may be carried out or completed outside the practical sessions and must be handed in by the end of the semester at the latest. This work is optional, but you will be awarded additional points if it is completed.

1 First part (mandatory) : algorithms on graphs using NetworkX

In this first part, we will be working on the two graphs constructed during previous practical lab (practical lab - session 1), based on the networks represented on the boards of the games "Pandemic Hot Zone North America" and "Ticket to ride - London". You will need to use NetworkX functions to answer to the questions. It is up to you to use the NetworkX's documentation to identify the right functions to use to answer to each given problem.

Required work :

1. How do you transform the graph of "Ticket to ride" into a simple graph ? Apply this operation to transform it into a simple graph. From now on, we will refer to this transformed graph for the "Ticket to ride" network.
2. Apply Depth First Search (DFS) graph traversal algorithm on the "Ticket to ride" graph, starting from the "British Museum" node. Do the same now starting from the "Globe Theatre" node.
3. Determine the shortest path between the nodes "Seattle" and "Santo Domingo" in the "Pandemic" graph. Check the result by applying Dijkstra's algorithm by hand.
4. Apply the appropriate NetworkX function to determine the minimum-weight spanning tree of the two graphs. For each graph, is there a single minimum-weight spanning tree ? Check your results by applying Prim's and Kruskal's algorithms by hand.

2 Second part (optional) : implementation of algorithms on graphs with Python

Implement in Python the Depth First Search algorithm and the Dijkstra algorithm. You can use the data structures and functions proposed by NetworkX (without, of course, directly using the functions for depth first search traversal and shortest path search), or use your own data structures that you may have implemented during the first practical lab session.