FRENCH-AZERBAIJANI UNIVERSITY
UE709 Network Algorithms L3/S5
PW1 - Introduction to Graph Theory
October 2019

Working with Python and NetworkX

1 Requirements

This laboratory uses Python programming language. Install the following modules for Python3:

- `pandas`
- `matplotlib`
- `networkx`

Download both `cities_in_az.csv` and `airports.csv` datasets available in this course at moodle3.unistra.fr.

2 Some information on NetworkX

i. To convert data from a dataframe into a graph:

```
import pandas as pd
import networkx as nx

data = pd.read_csv('mydatafile.csv')
print(data.head()) # to see first rows of the dataframe
print(data.columns) # to see all column titles of the dataframe
G = nx.from_pandas_edgelist(data, source='X', target='Y', edge_attr=True)
```

where `X` and `Y` are the titles of columns that will be considered as the endpoints of each edge (row). Other columns are data for each edge (`edge_attr=True`).

ii. To aggregate data to nodes:

```
def attribute_for_nodes(G, attribute, default_value):
    """
    Create an attribute for every node in G and set it with default value;
    If called again, reset all nodes' attribute to default value
    """
    for g in G.nodes.keys():
        G.nodes[g][attribute] = default_value
```

iii. About nodes and edges:

- `G.nodes()` returns a list with every node of the graph G
- `G.nodes[node_identifier]` returns a dictionary with all attributes for the `node_identifier` of G
- `G.nodes[node_identifier][attribute]` returns a dictionary with all attributes for that node
- `G.edges()` returns a list of tuples constituted by the endpoints of all unique edges (undirected) of G
- `G.edges[node_id1, node_id2]` returns a dictionary with all attributes for the edge `(node_id1, node_id2)` of G
- `G.edges[node_id1, node_id2][attribute]` returns the value of the attribute for that edge `(node_id1, node_id2)`
- `list(G.neighbors(node_identifier))` returns a list with all nodes that share an edge with the `node_identifier` (don't need to convert to list if you need to iterate through it)

iv. To visualize the graph

```
import matplotlib.pyplot as plt

plt.figure()
nx.draw_networkx(G, with_labels=True)
plt.show()
```

v. To find more information about NetworkX:
https://networkx.github.io/documentation/latest/index.html

3 Activities

3.1 Cities

A very basic dataset is available in `cities_in_az.csv`. Use columns `Origin` and `Destiny` as the endpoints of your graph edges. Then:

1. Plot the graph to visualize it
2. Create a function to find anyone **path** from a given origin to a given destiny (not supposed to be optimal) - you could get inspiration from Depth-First Search (DFS) algorithm: https://en.wikipedia.org/wiki/Depth-first_search
3. Create a function to evaluate how many hours the achieved path take to be completed

Is the found path optimal? Try to verify this comparing your results to different implementations of the path function.

3.2 Airports

The `airports.csv` dataset have a sample of flights from the USA. The below variables have been provided:

- Origin and destination
- Scheduled time of arrival and departure
- Actual time of arrival and departure
- Date of the journey
- Distance between the source and destination
- Total airtime of the flight

Use columns `Origin` and `Dest`, destinations of flights, as the endpoints of your graph edges. Then:

1. Plot the graph to visualize it
2. Use the same function you implemented in the previous activity to find anyone **path** from a given origin to a given destiny
3. Create a function to evaluate the distance (column `Distance` of the dataset) the achieved path take to be completed
4. Create a function to evaluate the time (column `AirTime` of the dataset) the achieved path take to be completed

Compare your results to different implementations of the path function.