University of St. Gallen

Institute of Computer Science

Fundamentals of Computer Science (3,125), HS2019
I. Mizutani, J. Sales, B. Bermeitinger, S. Mayer, B. Weber
iori.mizutani@unisg.ch

Assignment 4: Graph Network Analysis

Deadline: Oct 15, 2019; 12:00 CET

Introduction A *graph* is a powerful data structure to express the relations between things by connecting nodes (vertices) with edges (paths), and it allows you to extract useful information from the data set as well as to discover characteristics of the network. In this assignment, you will work on two data sets: movies and flight information.

Rationale You will learn how to use a full-blown Python package for graph representation and manipulation as well as Python Dictionary datatype effectively.

Prerequisites Basic understanding of graph data structure, graph network analysis methodologies, and NetworkX[1] Python package. Install the NetworkX package in your environment. In Thonny, use the menu [Tools] → [Manage packages...], type "networkx", and click [Install]). You **ONLY** need to edit the following files:

- `movie_analyzer.py`: The template file for Task 1. The `main()` function is already provided to test your code.
- `a4_task2/airline.py`: the file describing the class `Airline`. You only need to implement the instance method `get_route(self, source, target)`.
- `airline_analyzer.py`: The template file for Task 2. The `main()` is already provided to test your code.

Note: This assignment contains data sets (`movies.txt`, `airlines.dat`, `airports.dat`, `routes.dat`) in the package. The template files already provide functions to load them into a `networkx.Graph` object (Task 1) and dictionaries of `networkx.Graph` objects (Task 2).

(1) Movie Stars (5 Points)

In this task, you will work on a movie data set [2], which contains a list of movies and the performers who appeared in them. Each line gives the name of a movie followed by the cast (a list of the names of the performers who appeared in that movie).

The function `load_movies_as_graph()` in the file `a4_task1/movie.py` reads the `movie.txt` file and loads the data into a `networkx.Graph` object. Each node is either a `string` (name of performer) or a `Movie` object.

(a) Study the `load_movies_as_graph()` function in `a4_task1/movie.py` and understand how the movie data is added as nodes/edges in the graph. There are also some "magic methods" [3] in the file, but you do not have to go through them. Read the documentation for the

[1] https://github.com/networkx/networkx

[2] Originally from Sedgewick's book: https://introcs.cs.princeton.edu/python/45graph/movies.txt

[3] The methods whose identifiers are `__double_leading_and_trailing_underscore__`: https://docs.python.org/3/reference/datamodel.html#basic-customization

`networkx` functions[4] and `networkx.Graph` class methods[5] definitions. You can already run `movie_analyzer.py` to see some information about the graph. (Nothing to submit)

(b) Read the docstring of the function `get_list_of_performers(g, movie)` and implement the function. Hint: use the `nx.all_neighbors()` function[6] which returns an iterator[7]. After completed, uncomment the comment block for Task 1(b) in the `main()` function and run `movie_analyzer.py` to test the function.

(c) Read the docstring of the function `get_list_of_movies(g, performer)` and implement the function. After completed, uncomment the comment block for Task 1(c) in the `main()` function and run `movie_analyzer.py` to test the function.

(d) Read the docstring of the function `get_number_of_appearance_dict(g)` and implement the function. Hint: use the built-in function `isinstance()` to identify if a node is a `Movie` object or a `string` of performer's name. After completed, uncomment the comment block for Task 1(d) in the `main()` function and run `movie_analyzer.py` to test the function.

(2) Airlines (5 Points)

In this task, you will work on the *OpenFlights Airports Database*[8] which consists of three files: `airlines.dat`, `airports.dat`, and `routes.dat`. An "OpenFlights identifier" is assigned to each airline in addition to the IATA airline code [9] to differentiate subsidiaries (e.g., "Lufthansa"(id: 3321) [10] and "Lufthansa Cargo"(id: 3320) have the same code "LH"). In the `main()` function, two dictionaries are already defined:

- `airports_dict`
  - Key: Airport Code string (e.g. `"ACH"`)
  - Value: Name string of the airport (e.g. `"St Gallen Altenrhein Airport"`)
- `airlines_dict`
  - Key: OpenFlights identifier integer (e.g. `4559`)
  - Value: `Airline` object
    (e.g. `name: "Swiss International Air Lines", code: "LX", graph: <nx.Graph>`).

(a) Study the class `Airline` in the file `a4_task2/airline.py` except for the "magic" methods. You'll see this class has a `graph` attribute to store an instance of `networkx.Graph`, in which the nodes represent the airports and the edges as the routes (i.e., flight paths) – a route is defined as a connection between two airports. You can already run `airline_analyzer.py` to see some output from the examples. (Nothing to submit)

(b) Read the docstring of the instance method `get_route(self, source, target)` in `a4_task2/airline.py` and implement the function. After completed, uncomment the comment block for Task 2(b) in the `main()` function of `airline_analyzer.py` and run `airline_analyzer.py` to test the function.

[4] https://networkx.github.io/documentation/stable/reference/functions.html
[5] https://networkx.github.io/documentation/stable/reference/classes/graph.html#reporting-nodes-edges-and-neighbors
[6] https://networkx.github.io/documentation/stable/reference/generated/networkx.classes.function.all_neighbors.html#networkx.classes.function.all_neighbors
[7] https://www.programiz.com/python-programming/iterator
[8] https://openflights.org/data.html
[9] https://en.wikipedia.org/wiki/Airline_codes
[10] The route information is missing from the dataset.

(c) Read the docstring of the function `get_airlines_per_route(airlines_dict, source_airport_code, target_airport_code, max_hops)` in `airline_analyzer.py` and implement the function. After completed, uncomment the comment block for Task 2(c) in the `main()` function and run `airline_analyzer.py` to test the function.

(d) Read the docstring of the function `find_most_connected_airport(airlines_dict)` in `airline_analyzer.py` and implement the function. After completed, uncomment the comment block for Task 2(d) in the `main()` function and run `airline_analyzer.py` to test the function.

Hand-in Instructions Create a file `README.txt` that includes 1.) any reference that you used to complete this assignment, 2.) pitfalls you encountered and 3.) short explanations of your solution if necessary. Fill in your name and student ID in the commend header of files you edited.

Compress the whole folder with the Python files, the data files, and the `README.txt` to a `zip` file named "`assignment4.zip`". Upload the zip file to your exercise group's Course page on Canvas. See Figure 1 for the list of files should be included in the zip file.

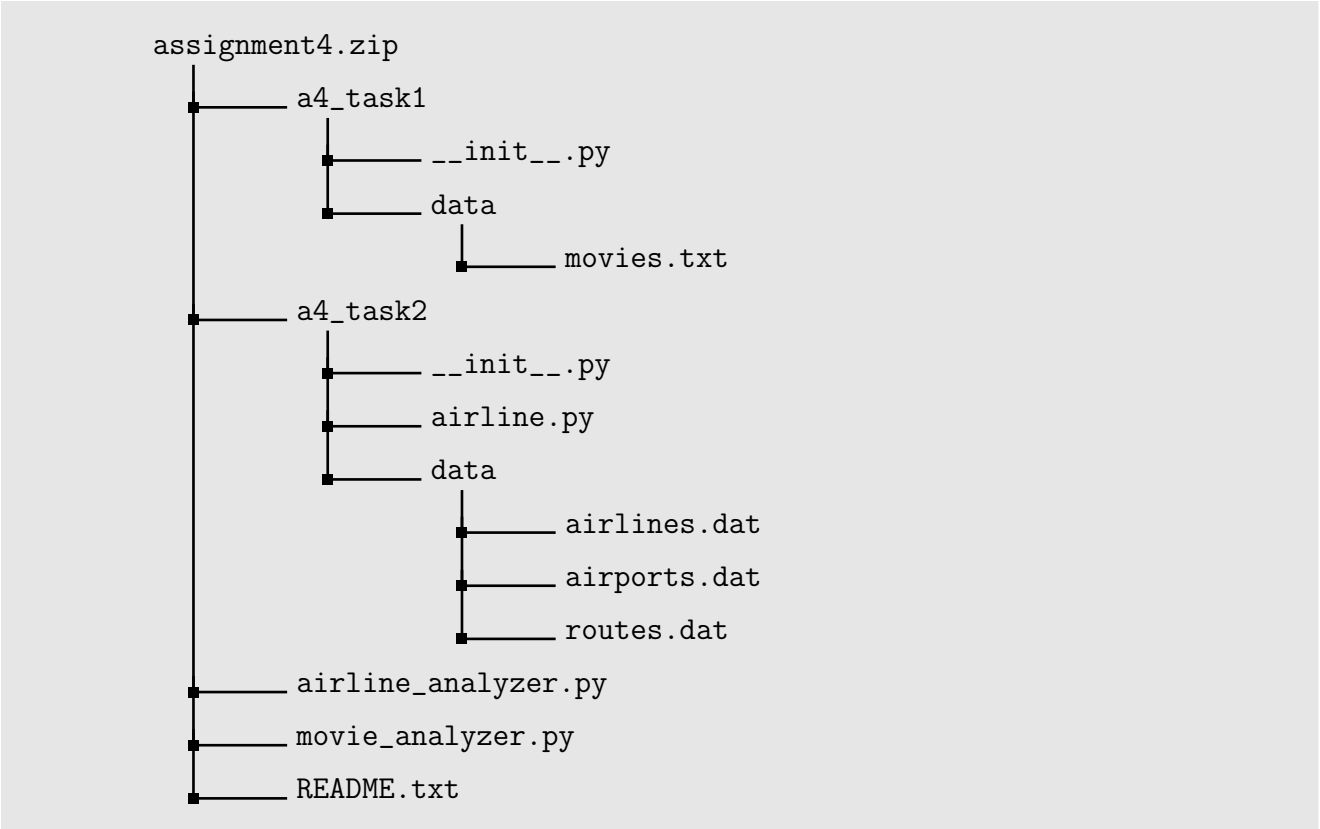

Figure 1: The required files for submission.