Data Structures and Representation (ZEIT2103)

Assignment 3 – Social Network

100 Marks = 27%

Due Date: 11:55 PM, Friday 23 May 2025

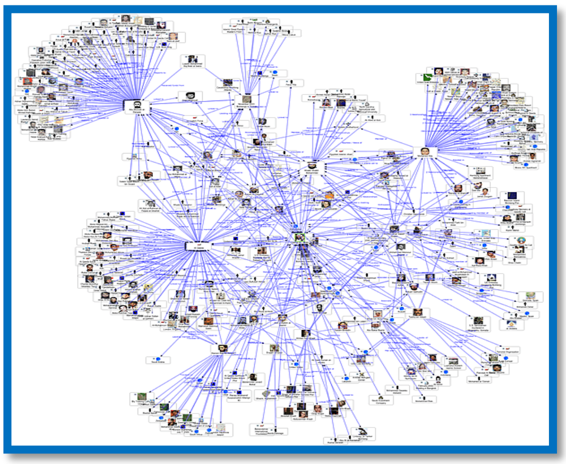
Assignment Summary

In this assignment, you will investigate using graphs, hashing, reading files, priority queues and other data structures and dealing with multiple interacting objects.

This is **a group assessment** (no more than four students in each group). Although a single copy of the code is needed for the submission, the project will be marked as a project score for every team members. Also, each team member will get an individual mark for their contribution and Task 6 (reflection). Please fill your group members in ZEIT2103_DSR_ass2_Group.xlsx

Problem Description

A social network can be represented as an undirected graph with each person represented as a node and each connection between two nodes an edge.



In this assignment, you will be given a randomly generated sample of data consisting of more than 1,000 accounts (people) (data.txt), with information of each, i.e., id, name, date of birth (DoB), suburb and a set of numbers representing any connection between the current person and the other people with the id provided, provided in the supplied file. In the figure below, as an example, Gillian Garnett, whose DoB is 27 November 1994 and the suburb where he lives Majura, is connected to Minna Whittaker and Stephen Ernest, whose IDs are 1 and 3, respectively.

```
1       Minna Whittaker 1980-06-15, Majura, 2
2       Gillian Garnett 1994-11-27, Majura, 1, 3
3       Stephen Ernest  2006-11-01, Canberra Central, 2
```

You will be asked to read this data and represent it as a graph. Bearing in mind the amount of data, as hashing will be used to accelerate the insertion and search processes, you will be required to investigate the efficiency of different hashing functions. Then, you will implement your *mini social network application,* which suggests friends to a given account, finds mutual friends of two friends and presents reminders of upcoming birthdays.

Note: Please read task 6 to get an overall idea about the expectations in terms of teamwork, for example:

- effective composition and distribution of tasks (you can use MS Teams[1], Trello or other tools),
- clear milestones are identified,
- all team members are contributing,
- regular meetings (at least once a week) - think of having an agenda for each meeting;
- managing the source code: you may use Git[2][3][4][5] or any other similar tool; not mandatory, though;

Getting Started – Resources

Some resources have been provided for this assignment in the assignment area of the course Moodle website. They consist of the following class definitions:

- `NodeInterface.py` –is an abstract base class that holds several methods you need to implement in the Node.py class.
- `Node.py` – represents a node (person) in the graph with its adjacency list of edges (friends).
- `GraphInterface.py` –is an abstract base class that holds the basic methods needed to implement the Graph.py class
- `Graph.py` that constructs an undirected graph with some basic operations
- `SocialNetworkInterface.py` contains some basic methods needed to complete the SocialNetwork.py class
- `HarnessClass.py` class, which you will update by implementing your test cases in it.
- `data.txt` – contains randomly generated accounts (people), with information of each, i.e., id, name, date of birth (DoB), suburb and a set of numbers representing any connection between the current person and the other people with the id provided;

Assignment Tasks

In general, this assignment involves several kinds of tasks: implementing some classes, documenting design decisions and reflecting on your programming processes. The latter two should be included in the pdf file ass2.pdf, which should be structured with a heading for each task and sub-headings for the different activities. Do not forget to mention the group members in the report.

[1] https://support.microsoft.com/en-us/office/create-a-plan-with-planner-in-teams-fa65ee5c-3c9b-42da-97b3-2fcd1a1c626d

[2] https://www.youtube.com/watch?v=8JJ101D3knE

[3] An example of using Git in VS Code https://code.visualstudio.com/docs/sourcecontrol/overview

[4] https://www.youtube.com/watch?v=z5jZ9lrSpqk

[5] https://www.jetbrains.com/help/pycharm/using-git-integration.html

Task 1. Implementing `Node` and `Edge` Classes (10 Marks)

- As previously mentioned, each person in a graph is represented by a node. Therefore, the `Node` class (which implements `NodeInteface`) has four main attributes: `id`, `name`; `dateOB` (of type LocalDate); `suburb`; and `adj`, which represents connections to friends and is structured as a HashMap, whereby `HashMap < Integer, Edge> adj`.

  You need to

  - define the constructor (`Node(id: int, name: str, dob: datetime.date, suburb: str)`) that defines the initial values of personal information using the supplied `id`, `name`, `doB` and `suburb`. Also, remember to complete the declaration of the adj variable. **[1 mark]**
  - write all the getter methods required **[2 marks]**, and
  - implement the `__str__()` method, which returns a string representation of all data about a Node object; the string format may look like `Node {id = 1, name = Minna Whittaker, DOB = 1980 -06-15, suburb = Majura}` **[2 marks]**

  Expectations for documentation [2 marks]: (1) documentation of your test cases as instructed in lectures; (2) a UML diagram;

- As the `Edge` class has one attribute friend, that is,(`friend: Node;`), you should
  - implement the constructor (`Edge(friend: Node)`) that creates a new edge with the supplied parameter (friend); **[1 mark]**; and
  - write the `__str__() method`, which returns a string representation of all data about an `Edge` object; the string format may look like this: `friend= {1, Minna Whittaker, 1980 -06-15, Majura}` **[1 mark]**;

  Expectations for documentation : (1) documentation of your test cases and (2) a UML diagram; **[1 mark]**

Note: Although no exceptions are expected to be thrown in this task, you can do so if you see needed.

Task 2. Hashing: overriding `__hash__()` and `__eq__()` methods in Node Class (8 Marks)

This task requires you to write a hash function designed specifically for Node objects, overriding the __hash__() method **[3 marks]** in the Node class. When implementing it, you can choose which attributes to use, how to normalise and weigh them, etc. You may try some of the hash functions presented in lectures or conduct some research when designing yours.

Warning 1: you should NOT use the built-in hash() method for any part of the calculation of your hash function. If you do, you will automatically receive a ZERO mark for this task.

Then, to conform with the attributes you used in `hash()`, you need to override the `__eq__(other: object)` method **[2 marks]**, as discussed in the lecture.

Expectations for documentation: (1) at least one paragraph (and no more than half a page) describing your approach while rationalising/explaining any special weightings adopted or attributes omitted **[4 marks]**. You should reference any literature you use in ass2.pdf; (2) test the hash() method you implemented **[1 mark]** (i.e., check how many duplicate keys are generated (you can use MS Excel to see if there are duplicates or insert all values in a set and check its size).

Note: Although no exceptions are expected to be thrown in this task, you can do so if needed.

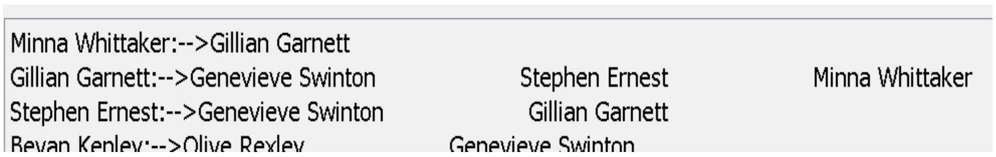
Task 3. Implementing `Graph` Class (23 Marks)

To begin forming an undirected graph representing a social network, a `Graph.py` class is provided to you, which has an empty constructor, and an attribute called `nodeList` will hold all nodes (people information) in the graph.

Then, implement the following:

- `add_node(node_id: int, name: str, dob: datetime.date, suburb: str)`, which creates a new Node based on the data provided, then adds it to a graph if it does not exist, and returns the Node created; otherwise, throw an exception; **[3 marks]**
- `add_edge(from_node: Node, to_node: Node)` creates and adds an edge between both nodes if it does not exist; otherwise, throw an exception; remember, this is an undirected graph, so you need to add the `Edge` to both nodes. **[3 marks]**
- `remove_edge(from_node: Node, to_node: Node)`, which removes an edge between two nodes if it exists. No action is needed if the edge does not exist. **[3 marks]**
- `remove_node(node: Node)` appropriately removes a node from nodeList and any edge between this node and other nodes; **Hint**: if a node does not exist, throw an exception; **[3 marks]**
- `get_neighbors(node: Node)` returns a set of friends of a given node; otherwise, throw an exception if the node does not exist. **Hint:** returning the set of friends can be done with only one line of code and **[3 marks]**
- `__str__()`, which returns a string representation of each person's name and friends. A possible format of this string may be **[3 marks]**
  - person 1: --> friend1 friend2

- person 2: --> friend1 friend2



```
Minna Whittaker:-->Gillian Garnett
Gillian Garnett:-->Genevieve Swinton          Stephen Ernest          Minna Whittaker
Stephen Ernest:-->Genevieve Swinton           Gillian Garnett
Bevan Kenley:-->Olive Rexley             Genevieve Swinton
```

Expectations for documentation : (1) documentation of test cases for each method; (2) justification of any helper method(s) you might implement; (3) UML diagram; **[7 marks]**

Task 4. Implementing `SocialNetwork` Class (14 Marks)

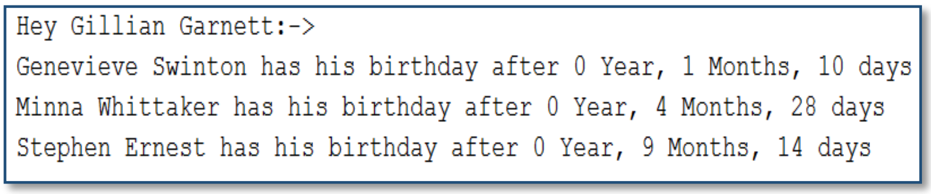
This task aims to read the data files provided to you, construct a graph, and conduct some types of social media operations. This class has an attribute sn of type `Graph` instantiated in the constructor. Then, implement:

- `process_file()` that reads the data file and builds the graph accordingly while remembering to call this method in the constructor; **[5 mark]**
- `suggest_friends(current_person: Node)` which returns a list of suggested friends. This will return a `List<Node>` such that each Node in the list is a friend of a friend that lives in the same suburb as `current_person`; **[5 mark]**
- `get_mutual_friends(x: Node, y: Node)`, which returns a list of all names of mutual friends of two friends (x and y). **[4 mark]**

Expectations for documentation : (1) a short paragraph justifying the selection of the class you used to read data.txt **[1 mark];** (2) UML diagram **[1 mark];** (3) justification of your approach to addressing this task **[1 mark]**; and (4) big O discussion for the suggestFriends **and** getMutualFriends methods**[3 mark]**;.

Task 5. Implementing Birthday Reminder (no assistance given) (15 Marks)

Extend the `SocialNetwork` class by writing a `remind_bd_events(current_person: Node)` method that takes a `Node` object as an input and returns a String that represents all the friends of the input node sorted based on the periods until their date of birth (dob); for instance, the figure below, which was generated using a different database, shows all the friends of Gillian Garnett sorted based on their next birthdays.



```
Hey Gillian Garnett:->
Genevieve Swinton has his birthday after 0 Year, 1 Months, 10 days
Minna Whittaker has his birthday after 0 Year, 4 Months, 28 days
Stephen Ernest has his birthday after 0 Year, 9 Months, 14 days
```

[15 marks]

Hint: (1) use PriorityQueues; and (2) you will need to override the compareTo method in Node class; this is to help with sorting based on the period remaining until their birthday

<u>Expectations for documentation</u>: (1) justification of your design; (2) documentation of test cases for each method; [**5 marks**]

Task 6. Reflection on Teamwork (30 Marks) (including Individual Contribution 15 Marks + Individual Reflection 15 Marks)

You should reflect on

- How were tasks distributed (who did what, contribution distribution in your team)? Did the team meet the project schedule? The tool(s) you used to manage the project, how did you maintain team communication (i.e. how often did you meet and how meetings were organised), how you managed the source code (i.e., did the team use any version control tools, like GitHub or similar tools, not mandatory, though), the challenges the team faced and how you dealt with them.
- Did you encounter any technical or personal challenges resulting from teamwork? How did you deal with them? What could you do to prevent these challenges in your future teamwork projects?
- Please attach your own code (only parts you achieved) to the appendix of your documentation.
- Have your Python skills increased? What lessons did you learn?
- Any other points you want to highlight.

There is no exact expectation about the length of this task, but you need to answer all points.

Marking and Submission

You are required to refer to and follow the **'DSR Programming Assignment Rubric'** document. Please take a look at the marking scheme and the rubric attached to this assignment on Moodle.

When you have completed your work, you must submit the Python files you wrote through Moodle as a zip file. However, the ass1.pdf should not be zipped.

Use of Generative Artificial Intelligence (AI), such as ChatGPT

As this assessment task involves some planning or creative processes, you are permitted to use software to generate initial ideas. However, you must develop or edit those ideas to such a significant extent that what is submitted is your own work, i.e. only occasional AI generated words or phrases may form part of your final submission. It is a good idea to keep copies of the initial prompts to show your lecturer if there is any uncertainty about the originality of your work.

If the outputs of generative AI, such as ChatGPT, form a part of your submission, it will be regarded as serious academic misconduct and subject to the standard penalties, which may include 00FL, suspension and exclusion.

* To cite: OpenAI (Year Accessed). ChatGPT. OpenAI. https://openai.com/models/chatgpt/

* Please note that the outputs from these tools are not always accurate, appropriate, nor properly referenced. You should ensure that you have moderated and critically evaluated the outputs from generative AI tools such as ChatGPT before submission