Introduction to Java (BIOL60201)

A network analysis programme

Networks are used to represent many different types of biological data, including genetic regulation, protein-protein interactions, drug-target associations, etc. In this project, you will construct a Java programme allowing users to create and analyse biological networks.

Networks are formed by a set of *nodes* connected by *edges*. Each edge connects two nodes, representing an interaction between them (Figure 1). Edges can be directed, which means that the interaction is known to go from a source to a target, or undirected, which means that the interaction has no specified direction. In this project we will assume that our networks are undirected. Note that self-connections are generally allowed, for example a protein can interact with itself to form a dimer.

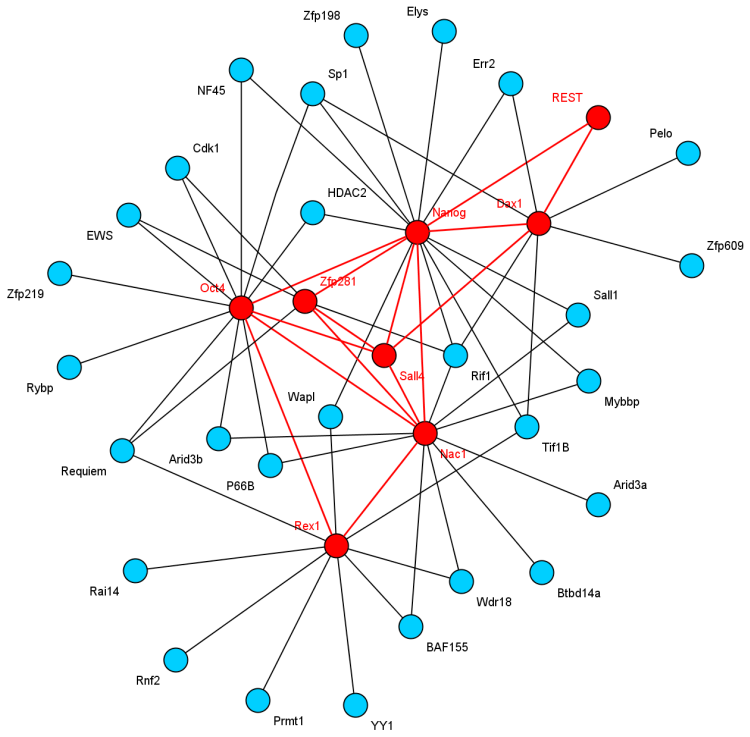

Figure 1: Example of protein interaction network.

Step 1: Node and Edge classes

a) Create a Node class representing a single node of a network. The Node class should contain a String variable containing the name of the node. It should have a default constructor that creates a node with an empty name, and another constructor with a String argument that creates a node with the given name. The Node class should also have a method that returns the name of the node.

b) Create an Edge class representing an edge connecting two nodes. The Edge class

needs no default constructor but should have a constructor accepting two Node arguments that creates an edge connecting these two nodes.

Step 2: The Network class

a) Create a Network class representing an undirected network. The Network class should contain a list of nodes and a list of edges. Write a default constructor for the Network class that creates an empty network.

b) Write a method for the Network class that reads a file containing a list of protein interactions and creates a Network out of them. The file *PPInetwork.txt* is provided as an example: each line in this file represents an interaction between two proteins; these proteins are named after their UniProt identifiers; both names are separated by a tabulation. Duplicate interactions should only be included once. Self-interactions (a protein interacting with itself) should be included.

c) Write a method for the Network class that allows the user to manually add a new interaction to an existing network. This method should not allow the addition of an interaction that is already contained in the network (remember that the network is undirected).

Step 3: Network analysis

The *degree* or *connectivity* of a node in a network is the number of edges connected to the node.

a) Write a method that allows the user to enter a node name and returns the degree of the corresponding node.

b) Write a method to calculate the average degree of all nodes in the network.

c) Write a method to find the *hubs* of the network, which are the nodes of highest degree. The method should print the value of the highest degree and the names of all nodes having that degree.

d) Write a method enabling the user to save the full *degree distribution* of the network in the form of a table. The first column of the table should contain the degree and the second column the number of nodes having that degree.

Step 4: Main programme

Your main programme should enable the user to read the *PPInetwork.txt* file provided, add new interactions, print the average degree and the names of its hubs, and create an output file containing the degree distribution. The programme should preferably have a user-friendly Graphical User Interface and handle possible input errors by the user or in the data.

Assessment

The assessment will be based on a report and an evaluation of the software produced. The report and software should be the result of your individual work.

Your report should be structured in the form of a short research article (maximum 1500 words) as follows:

- The *Abstract* should summarise the purpose and features of the software.
- The *Introduction* should explain the motivations and aims of the software.
- The *Methods* should describe the class structure and functionalities of the software.
- The *Results* should present the results obtained using the data file provided.
- The *Discussion* should critically assess the software and compare it with other tools.

Your source code files should contain the Node, Edge and Network classes, as well as a class containing your main programme. The main programme should be designed to be run by an inexperienced user without need to edit the source code. If any additional libraries are needed to run your code these must be provided with your submission, together with instructions for running.

50 % of the mark will be awarded for the report and 50 % for the quality of the software.

All files should be submitted via Blackboard as an archive (zip) file with your name.

Submission deadline: Thursday 11th February 2021 at 16:00 (UK)