COSC 2P03 Advanced Data Structures: Assignment 4

Instructor: Yifeng Li*[1]

[1]Department of Computer Science, Brock University

November 25, 2022

1 DrugBank and Drug Design

In this assignment, we continue to play with the DrugBank data to build a graph. You are expected to finish the following tasks.

2 Your Tasks (Total: 12 marks)

You should define classes named `Vertex` and `DrugGraph` with the following requirements (feel free to define extra variables, classes, and methods if needed).

1. You should define class `Vertex` with data attributes `drugBankID`, (this is used as node label), `genericName`, `SMILES`, `url`, `drugGroups`, and `score`, `wasVisited` (or `known`), `dist`, and `path`. This class has at least one method, named `displayDrug` to print out the information on the screen. **(0.5 mark)**

2. Define a method named `readData` under the `DrugGraph` class to realize the following functionality:

   - Load all the provided information from the given text file `dockedApproved.tab` to an array variable named `vertices` which is an attribute of the `DrugGraph` class. Each element in array `vertices` is an object of class `Vertex`. You should give reasonable initial values to variables `wasVisited`, `dist`, and `path`. **(0.5 mark)**
   - Load the provided data from the given text file `sim_mat.tab` to an array variable named `W` which is also an attribute of the `DrugGraph` class. Note that, the data in `sim_mat.tab` is the *similarity* matrix of the drugs, where the value at the i-th row and the j-th column is the Tanimoto similarity between the i-th drug and the j-th drug; the row and column orders of drugs are same as the drug order in file `dockedApproved.tab`. Since the values in `sim_mat.tab` are similarities, we need to convert these values into dissimilarities (i.e., distances) in order to have proper values in array `W`. Suppose a value in the file is denoted by v_{ij}, the corresponding value in matrix `W` should be

   $$w_{ij} = \begin{cases} 1 - v_{ij} & \text{if } (1 - v_{ij}) \leq threshold \text{ and } i \neq j \\ \infty & \text{otherwise} \end{cases}, \tag{1}$$

   where $threshold \in (0, 1)$. We use $threshold = 0.7$ in this assignment. Here `W` is actually an weighted adjacency matrix (in terms of dissimilarity/distance). **(1 mark)**
   - Based on `W`, obtain the discrete (unweighted) adjacency matrix `A` which is an array variable under class `DrugGraph`. A value a_{ij} in matrix `A` is calculated as

   $$a_{ij} = \begin{cases} 1 & \text{if } w_{ij} \neq \infty \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

   which means that matrix `A` is a 0-1 matrix. **(0.5 mark)**

*E-mail address: yli2@brocku.ca

- Note that, you should not use the Brock basic IO package anymore, otherwise, 1.5 marks will be deducted.

3. Define a method named `BFS(int i)` for breath-first search on a graph starting from a drug index `i`. **(1 mark)**

4. Define a method named `findModules` under the `DrugGraph` class to find the number of disconnected components (module) in the graph. This method should print the number of vertices in each module in an sorted (descending) order. Hint: you may create a variable of integer type named `module` under class `Vertex` to record the module ID for a drug. Within this method, you should repeatedly call the `BFS` method to traverse through unvisited vertices. For each BFS call, the traversed drugs are given the same module ID (e.g., one of 0, 1, 2, ...). Module 0 should be larger then Module 1, and so on. That is, Module 0 is the largest module, Module 1 the second, ... **(1.5 mark)**

5. Define a method named `keepAModule(int moduleID)` that keeps the connected component with a given module ID and removes all other modules. Note that, all the methods defined below are based on the kept module rather than the entire graph which is disconnected. **(1 mark)**

6. Define a wrapper method named `findShortestPath(fromDrug, toDrug, method)` to find the shortest path from `fromDrug` to `toDrug` on the connected module, where `fromDrug` and `toDrug` are the labels of two drugs. `method` is either ``` ``unweighted'' ``` or ``` ``weighted'' ```. This method should print the sequence of DrugBankIDs of the shortest path on the screen. To realize this wrapper method, you may need to define two extra methods to implement the unweighted algorithm (BFS-based algorithm) and weighted algorithm (Dijkstra's algorithm) for finding the shortest path starting from a vertex. **(3 marks)**

7. Define a method named `MSTPrim` to find the minimum spanning tree on the connected module. In this method, you should write the list of DrugBank IDs (can the same order as the drugs in your array attribute) on the MST to a text file named `MSTPrimResult.txt` with each row for a DrugBank ID and associated `dist` value. This function should also print the total length (i.e., the sum of `dist` values on the MST) of the MST on screen. **(2 marks)**

8. In the **main** function, instance (named `dg`) of the `DrugGraph` should be created. The following methods of this instance should be called sequentially: **(0.5 mark)**

   (a) `dg.readData(...)`

   (b) `dg.findModules(...)` (Hint: there are 87 modules, the first module has 1801 vertices.)

   (c) `dg.keepAModule(0)` This call is to keep the largest connected component only and remove other components.

   (d) `dg.findShorestPath('DB01050', 'DB00316', 'unweighted')`

   (e) `dg.findShorestPath('DB01050', 'DB00316', 'weighted')`

   (f) `dg.MSTPrim(...)` (Hint: the returned total cost should be 625.65.)

9. Your code should be well commented. **(0.5 mark)**

3 Submission

- Your source code.
- A PDF printout of your source code.
- Output text file `MSTPrimResult.txt` (from Task 7).
- A text file that contains all the printed out information on the screen/console.
- Compress the above files in a zipped folder named `COSC2P03_A4_Firstname_Lastname_StudentNumber.zip` and submit it through Brightspace before indicated due time.
- If any of the above require files are not submitted, 0 mark will be given to the whole assignment.
- Late submissions will not be accepted.

4 Academic Integrity

This assignment should be tackled individually. Outsourcing or teamwork is not allowed. Violation of this requirements will be seriously processed in accordance with university policies.