Work Integrated Learning Programmes Division
M.Tech (Data Science and Engineering)

(S1-21_DSECLZG519)
(Data Structures and Algorithms Design)
Academic Year 2020-2021

Assignment 1 – PS16 - [Vaccine Development] - [Weightage 12%]

1. Problem Statement

Since the advent of different viral strains, governments across the world are researching combining available vaccines to enhance immunity. Vaccines can be deployed after a gap of a few months to battle various strains. Assume that you are a global COVID vaccine researcher and you want to map which vaccines have been found effective against a virus strain (either in the past or present). For this you need to have some system of storing these vaccines and the strains they have neutralized. Assume that you have a list of N strains and M vaccines. For the sake of this assignment, let us assume that a particular vaccine could neutralize only two strains at max.

Write an application that maps COVID Strains and Vaccines and can answer the below queries:

1. List the unique strains and vaccines the researcher has collected in the system.
2. For a particular strain, help the reporter recollect the vaccines it has been neutralized by.
3. For a particular vaccine, list the strains that have been neutralized with it (past or present).
4. Identify if two vaccines neutralize similar strains. Vaccine A and vaccine B are considered to neutralize similar strains if they have been associated with the same strains (not necessarily at the same time or in the same year)
5. Can two vaccines A and B be connected such that there exists another vaccine C where A and C are neutralizing similar strains and C and B are neutralizing similar strains.

Requirements

1. Model the following problem as Graph based problem using Python 3.7. Clearly state how the vertices and edges can be modelled such that this graph can be used to answer the following queries efficiently.
2. Read the input from a file **inputPS16.txt**
3. You will output your answers to a file **outputPS16.txt**
4. Perform an analysis for the features above and give the running time in terms of input size: n.

The basic structure of the graph will be:

```
Class IMMUNIZATION:
        VaccineList=[] #list containing vaccine and strains
        Edges=[[],[]] #matrix of edges/associations
```

Operations:

1. **def readInputfile(self, inputfile)**: This function reads the input file **inputPS16.txt** containing the name of the strains and associated vaccines in one line. The name of the vaccine and strain should be separated by a slash.

   ```
   229E / CoviShield / Covaxin / SputnikV / Pfizer
   ```

   The function should create relevant vertices for the strains and vaccines and relevant edges to indicate the connection of a strain and its vaccines. Ensure that none of the vaccines or strains get repeated while creating the vertices of the graph.

2. **def displayAll(self):** This function displays the total number (count) of unique vaccines and strains entered through the input file. It should also list out the unique vaccines and strains. The output of this function should be pushed into **outputPS16.txt** file. The output format should be as mentioned below.

   ```
   --------Function displayAll--------
   Total no. of strains: 10
   Total no. of vaccines: 28
   List of strains:
   229E
   NL63
   OC43
   HKU1
   P1
   B117
   B1351

   List of vaccines:
   CoviShield
   Covaxin
   Pfizer
   SputnikV
   …….
   ```

3. **def displayStrains(self, vaccine):** This function displays all the strains a particular vaccine is associated with. The function reads the input strain name from the file **promptsPS16.txt** where the search id is mentioned with the tag as shown below.

   findStrain: CoviShield
   findStrain: Covaxin

   The output of this function should be appended into **outputPS16.txt** file. If a strain is not found, an appropriate message should be output to the file. The output format should be as mentioned below.

   ```
   --------Function displayStrain --------
   Vaccine name: Covaxin
   List of Strains:
   P1
   B117          (if strain is not found display appropriate message)
   -----------------------------------------
   ```

4. **def displayVaccine(self, strain):** This function displays all the vaccines associated with a strain. The function reads the input strain name from the file **promptsPS16.txt** where the search id is mentioned with the tag as shown below.

   listVaccine: P1
   listVaccine: B117

   The output of this function should be appended into **outputPS16.txt** file. If a vaccine is not found, an appropriate message should be output to the file. The output format should be as mentioned below.

   ```
   --------Function displayVaccine --------
   Strain name: P1
   List of Vaccines:
   Covaxin
   Pfizer
   CoviShield          (if vaccine is not found, display appropriate message)
   -----------------------------------------
   ```

5. **def commonStrain(self, vacA, vacB):** Use one of the traversal techniques to find out if two vaccine are related to each other through one common strain. The function reads the input vaccine names from the file **promptsPS16.txt** where the search id is mentioned with the tag as shown below.

   commonStrain: Covaxin : CoviShield

The output of this function should be appended into **outputPS16.txt** file. If a relation is not found, an appropriate message should be output to the file. The output format should be as mentioned below.

```
--------Function commonStrain --------
Vaccine A: Covaxin
Vaccine B: CoviShield
common strain: Yes, B117  (if no, display appropriate message)
-----------------------------------------
```

6. **def findVaccineConnect(self, vacA, vacB):** Use one of the traversal techniques to find out if two vaccines A and B are related to each other through a common vaccine C as defined in the question above. The function reads the input vaccine names from the file **promptsPS16.txt** where the search id is mentioned with the tag as shown below.

   ```
   vaccineConnect: Covaxin : Pfizer
   ```

   Display the entire relation that links vaccine A and vaccine B. The output of this function should be appended into **outputPS16.txt** file. If a relation is not found, an appropriate message should be output to the file. The output format should be as mentioned below.

   ```
   --------Function findVaccineConnect --------
   Vaccine A: Covaxin
   Vaccine B: Pfizer
   Related: Yes, Covaxin > P1 > CoviShield > B117 > Pfizer

   (if no, display appropriate message)
   -----------------------------------------
   ```

7. Add other functions that are required to perform the above minimum requirement

Sample file formats

Sample Input file

The input file **inputPS16.txt** contains names of the vaccines and its associated strains in one line. The name of the vaccine and strains should be separated by a slash (/).

Sample inputPS16.txt

```
229E / CoviShield / Covaxin / SputnikV / Pfizer
B1351 / CoviShield / Covaxin / SputnikV / Moderna
P1 / CoviShield / Covaxin / SputnikLight / CoronaVac
B117 / J&J / Moderna / SputnikLight / Pfizer
```

```
B1617 / CoronaVac / J&J / Moderna / Pfizer
L452R / CoronaVac / SputnikLight / SputnikV / Moderna
```

…………………..

Sample promptsPS16.txt

```
displayStrains: Covaxin
listVaccine: P1
commonStrain: Pfizer : Moderna
findVaccineConnect: Moderna : J&J
```

Sample outputPS16.txt

```
--------Function displayAll--------
Total no. of Strains: 6
Total no. of Vaccines: 9

List of Strains:
B117
P1
.
.

List of Vaccines:
J&J
CoviShield
…….
----------------------------------------
--------Function displayStrains --------
Vaccine name: Covaxin
List of Strains:
229E
P1
B1351       (if Strain is not found display appropriate message)
----------------------------------------

   ……
Rest of the function outputs.
```

Note that the input/output data shown here is only for understanding and testing, the actual file used for evaluation will be different.

2. Deliverables

1. Word document **designPS16_<group id>.docx** detailing your design and time complexity of the algorithm.
2. **[Group id]_Contribution.xlsx** mentioning the contribution of each student in terms of percentage of work done. Download the Contribution.xlsx template from the link shared in the Assignment Announcement.
3. **inputPS16.txt** file used for testing
4. **promptsPS16.txt** file used for testing
5. **outputPS16.txt** file generated while testing
6. **.py file** containing the python code. Create a single *.py file for code. Do not fragment your code into multiple files

Zip all of the above files including the design document and contribution file in a folder with the name:

[Group id]_A1_PS16_VaccineDevelopment.zip and submit the zipped file.

Group Id should be given as **Gxxx** where xxx is your group number. For example, if your group is 26, then you will enter G026 as your group id.

3. Instructions

1. It is compulsory to make use of the data structure(s) / algorithms mentioned in the problem statement.
2. Ensure that all data structure insert and delete operations throw appropriate messages when their capacity is empty or full. Also ensure basic error handling is implemented.
3. For the purposes of testing, you may implement some functions to print the data structures or other test data. But all such functions must be commented before submission.
4. Make sure that your read, understand, and follow all the instructions
5. Ensure that the input, prompt and output file guidelines are adhered to. Deviations from the mentioned formats will not be entertained.
6. The input, prompt and output samples shown here are only a representation of the syntax to be used. Actual files used to evaluate the submissions will be different. Hence, do not hard code any values into the code.
7. Run time analysis is to be provided in asymptotic notations and not timestamp based runtimes in sec or milliseconds.
8. Please note that the design document must include
   a. The data structure model you chose with justifications
   b. Details of each operations with the time complexity and reasons why the chosen operations are efficient for the given representation
   c. One alternate way of modelling the problem with the cost implications.

9. Writing good technical report and well document code is an art. Your report cannot exceed 4 pages. Your code must be modular and quite well documented.

Instructions for use of Python:

1. Implement the above problem statement using Python 3.7.
2. Use only native data types like lists and tuples in Python, do not use dictionaries provided in Python. Use of external libraries like graph, numpy, pandas library etc. is not allowed. The purpose of the assignment is for you to learn how these data structures are constructed and how they work internally.
3. Create a single *.py file for code. Do not fragment your code into multiple files.
4. Do not submit a Jupyter Notebook (no *.ipynb). These submissions will not be evaluated.
5. Read the input file and create the output file in the root folder itself along with your .py file. Do not create separate folders for input and output files.

4. Deadline

1. The strict deadline for submission of the assignment is **Wednesday, 22nd Dec, 2021.**
2. The deadline has been set considering extra days from the regular duration in order to accommodate any challenges you might face. No further extensions will be entertained.
3. Late submissions will not be evaluated.

5. How to submit

1. This is a group assignment.
2. Each group has to make one submission (only one, no resubmission) of solutions.
3. Each group should zip all the deliverables in one zip file and name the zipped file as mentioned above.
4. Assignments should be submitted via Canvas > Assignment section. Assignment submitted via other means like email etc. will not be graded.

6. Evaluation

1. The assignment carries 12 Marks.
2. Grading will depend on
   a. Fully executable code with all functionality working as expected
   b. Well-structured and commented code
   c. Accuracy of the run time analysis and design document.
3. Every bug in the functionality will have negative marking.

4. Marks will be deducted if your program fails to read the input file used for evaluation due to change / deviation from the required syntax.
5. Use of only native data types and avoiding libraries like numpy, graph and pandas will get additional marks.
6. **Plagiarism will not be tolerated. Copy / Paste's from web resources / or your friends' submission will attract severe penalty to the extent of awarding 0 marks. We will not measure the extent of such blatant copy pastes and details of who copied from whom and such details while awarding the penalties. It's the responsibility of the team to solve and protect your original work.**
7. Source code files which contain compilation errors will get at most 25% of the value of that question.

7. Readings

Text book: Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, Wiley (Students Edition). **Chapters:** 6