**(S2-20\_DSECLZG519) (Data structures and Algorithms Design)**

**Academic Year 2021-2022**

Software Design Documentation

**Assignment 1 - PS16- Group 16**

**Overview:**

This is a Traversal problem given a set of vaccine and strains we need to find the connected strains or vaccines.

**Given 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 vaccine

**Goal:**

The goal is to create a python program to

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.

**Design:**

The example given here has a set of strains and the vaccines that can enhance immunity for that strains

229E / CoviShield / Covaxin / SputnikV / Janssen

B1351 / CoviShield / Covaxin / Convidecia / Moderna

Here are taking all the vaccine and strains as nodes(vertices) and and relation between strain and vaccine as edges.

We traverse through the adjacency \_list create by above assumption using bfs to solve the details.

**Data Structure:**

We have chosen **list** as a data structure for storing edges , nodes and vaccineList.

vaccineList = [] *# list containing vaccine and strains*edges = [] *# matrix of edges/associations*

The graph we choose is adjacency list.

229E : ['CoviShield', 'Covaxin', 'SputnikV', 'Janssen']

CoviShield : ['229E', 'B1351']

Covaxin : ['229E', 'B1351']

The traversal mechanism we have chosen is BFS.

**Run Time Analysis:**

In our program we are using bfs traversal algorithm with adjacency list.

So, overall time complexity of the algorithm is O(V + E).

**Alternate Modelling:**

**Graph and traversal:**

**Instead of adjacency list we could have used adjacency matrix with BFS where time complexity would be same O(V+E) . while we could also have used DFS with both type of graphs and that would have given us O(V^2) time complexity.**

**Algorithm:**

|  |
| --- |
| **Input:** File inputPS16.txt contains total no. of use-cases, set of deadlines and bonus as space separated for every test case.  E.g.  No of use-cases: 2  Deadlines: 1 2 3 1 4  Bonus: 20 40 10 10 20  **Output:** File OutputPS5.txt lists the maximum bonus earned and the sequence of Jobs for every test case.  E.g.  Test case: 1  Maximum Bonus Earned: 90  Job Sequence: a b c e d  ------------------------------------------------ |