BLG 336E - Analysis of Algorithms II
2020/2021 Spring
Final Project
Question 4

- Please submit your homework only through Ninova. Late submissions will not be accepted.
- Please do not forget to write your name and student ID on the top of each document you upload.
- You should write your code in C++ language and try to follow an object-oriented methodology with well-chosen variables, methods, and class names and comments where necessary.
- Your code should compile on Linux using g++. You can test your code through ITU's Linux Server.
- Because your codes will be processed with an automated tool, `Calico`, make sure that your output format matches the given sample output.
- You may discuss the problem addressed in the homework at an abstract level with your classmates, but you should not share or copy code from your classmates or any web sources. You should submit your individual homework. In case of detection of an inappropriate exchange of information, disciplinary actions will be taken.
- If you have any questions, please ask on related message board named **Final Exam Q4 Questions** for this question.

1 Finding Bipartite Graphs (25 pts)

1.1 Problem Definition

There is an undirected graph with `n` nodes, each of which is numbered from `0` to `n - 1`. You are given a `2D` array graph, with `graph[u]` being an array of nodes adjacent to node `u`. The properties of the graph are as follows:

- There are no self-edges to be found: `graph[u]` does not contain `u`.
- There are no edges that are parallel: `graph[u]` does not contain duplicate values.
- If `graph[u]` contains `v`, then `graph[v]` contains `u`: the graph is undirected.
- The graph may not be connected, which means that there may be no path between two nodes `u` and `v`.

A graph is **bipartite** if its nodes can be divided into two independent sets A and B, with each edge connecting a node in set A and a node in set B.

*If and only if it is **bipartite***, return `True`.

1.2 Example 1

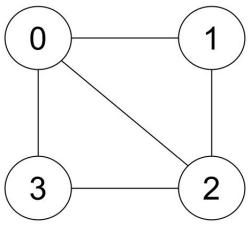

Figure 1: Input Graph

```
Input: graph = [[1,2,3],[0,2],[0,1,3],[0,2]]
Output: False
```

There is no way to divide the nodes into two distinct groups so that each edge connects a node in one group to a node in the other.

Illustration.

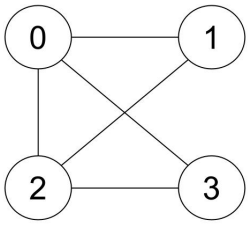

Figure 2: Input graph given in Figure 1 is shown NOT to be a bipartite graph.

1.3 Example 2

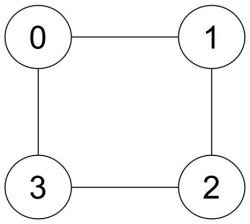

Figure 3: Input Graph

```
Input: graph = [[1,3],[0,2],[1,3],[0,2]]
Output: True
```

We can partition the nodes into two sets: {0, 2} and {1, 3}.

Illustration.

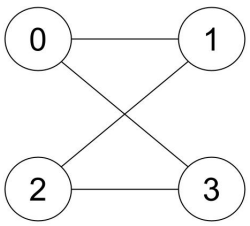

Figure 4: Input graph given in Figure 3 is shown to be a bipartite graph.

1.4 Implementation

In this question, you are expected to read graph data from file and store it in a 2D array. If necessary, you are allowed to use path finding algorithms such as Breadth-First Search, Depth-First Search, etc. Driver code is provided for convenience. Class declaration is also shown below.

```
class Solution {
public:
    bool isBipartite(vector<vector<int> >& graph);
};
```

Listing 1: Sample C++ code – Class declaration.

A driver code **q4.cpp**, a Calico test case file **q4.t** and two graph data files **q4_tc1.txt** and **q4_tc2.txt** are provided with the question. The driver code already includes class definition, and a function related to file reading and output formatting. Therefore, you only need to fill the empty `isBipartite` function.

Each test file includes a single graph data. Each line represents the adjacency list of a vertex. Each adjacency list includes items separated by a comma. For instance, neighbours of vertex `0` is listed as `1,2,3` in the data file.

You can compile and test your implementation with the below commands.

```
g++ -std=c++11 -Wall -Werror  q4.cpp -o q4
./q4 q4_tc1.txt
./q4 q4_tc2.txt
```

Please test your source files using Calico tool and make sure you pass given test cases before you upload your assignment. On your local computer, if you have already installed Calico, you directly use it from your terminal/console. On ITU's SSH server, you can use it as a Python module with `-m` option as shown below.

```
calico q4.t
python -m calico.cli q4.t
```