CS5130, Spring 2019, Dr. W. He

Project 3. Bipartite Graph Checking

(Due 5/19/2019 Sun, Optional for 3%)

Description:

In this project, we will process an undirected graph, not necessarily connected. The goal is to check if the given graph is *bipartite*. If it is a bipartite graph, you need to print out the vertices in two separate sets: V_1 and V_2. If it is not a bipartite graph, you need to find a triangle cycle in it.

The input part and the validation part are the same as that of Project 2. You can use that code directly.

Requirements:

1. (*Input*) The input comes from a text file that stores the *adjacency matrix* of a graph. Your program will take the file name of an input as a command-line argument. Then your program will read the content of the file line by line, in which each row corresponds to one row of the adjacency matrix of a graph.

2. (*Validation*) Before we process the graph, we need to validate the data to make sure that the given data corresponds to the adjacency matrix of a graph. Basically we need to check the following items:

   - (*Square Matrix*)

     Each row of the data file has a sequence of integers separated by a space character. Make sure that the number of rows equals the number of columns. Otherwise display an error message.

   - (*Bit Value Entries*)

     Check if each entry of the matrix takes the bit value: `0` or `1`. If not, display an error message.

   - (*No Self-Loops*)

     Since an undirected graph cannot have any self-loop, we need to check that all the diagonal entries must be `0`. Otherwise display an error message.

   - (*Valid Undirected Graph*)

     In order to make sure that this matrix corresponds to the adjacency matrix of an undirected graph, you need to check if it is *symmetric*. If not, display an error message.

3. (*Bipartite Graph Detection*) Note that your graph may contain multiple *connected components*. For each connected component, check if it is a bipartite graph. If it is, separate the vertices and than process the next connected component until you go through all the components.

   If you detect that one of the connected components is not bipartite, find a triangle cycle and you can stop processing.

4. (*Output*) In the case that it is bipartite, print out two separate sets. Only one solution is enough.

   In the case that it is not a bipartite graph, print out a triangle cycle with some message.

5. (*Testing*) You need to prepare your own testing files for your project development. I will use my own testing files for grading.