CSCI433 (Spring 19): Analysis of Algorithms
Project 1 **Friday, Mar. 8th**

This project has two parts. In the first part, you need to implement DFS in JAVA. Your program should take as input a graph adjacency matrix, and generate the following:

- Order in which vertices are first encountered. Assume that the vertices are visited in numerical order when no other order is specified by the search.
- Order in which vertices become dead-ends.
- The number of connected components in the graph.
- Tree edges.
- Back edges.

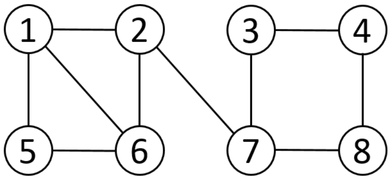

Figure 1: Example graph

In the example graph shown in Figure 1, the input graph adjacency matrix is specified in a text file with the following content:

```
0 1 0 0 1 1 0 0
1 0 0 0 0 1 1 0
0 0 0 1 0 0 1 0
0 0 1 0 0 0 0 1
1 0 0 0 0 1 0 0
1 1 0 0 1 0 0 0
0 1 1 0 0 0 0 1
0 0 0 1 0 0 1 0
```

Note that the vertex identifiers correspond to the row indices (and column indices) of the adjacency matrix. For the above example, DFS should generate the following vertex orderings (**count value for vertices**):

$$\text{First encountered}: 1 \quad 2 \quad 6 \quad 7 \quad 4 \quad 3 \quad 5 \quad 8$$

$$\text{First dead}-\text{ends}: 8 \quad 7 \quad 5 \quad 4 \quad 1 \quad 2 \quad 6 \quad 3$$

The number of connected components is 1. The tree edges and back edges are given by two adjacency matrices:
Tree edges

```
0 1 0 0 0 0 0 0
0 0 0 0 0 1 1 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0
```

Back edges

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix}$$

In the second part of the project, you need to implement BFS in JAVA. Your program should take as input a graph adjacency matrix as in DFS, and generate the following:

- Order in which vertices are first encountered. Assume that the vertices are visited in numerical order when no other order is specified by the search.
- The number of connected components in the graph.
- Tree edges.
- Cross edges.

For the graph in Figure 1, the program should generate the following vertex ordering (**count value for vertices**):

$$\begin{matrix} 1 & 2 & 6 & 8 & 3 & 4 & 5 & 7 \end{matrix}$$

The number of connected components is 1. The tree edges and cross edges are given by two adjacency matrices:
Tree edges

$$\begin{matrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Cross edges

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

Please send JAVA source code and executable to Blackboard before **Friday, March 8th, 2018**.