Graph Algorithms - Programming Assignment 2

Due: Thursday, October 12th and Friday, October 13th

Credit: This is a modification of a lab by Tim Kearns.

Directions: Your code will be due by 8pm Friday, October 13th on PolyLearn. You may use either Python 3.4.3 or Java 8.

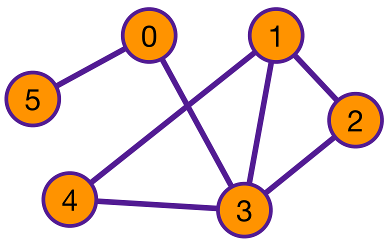
A (vertex) *coloring* of a graph is an assignment of color to each of the vertices such that no two adjacent vertices share the same color. See section 3.1 of your text for an example of a coloring of a map.

You will implement an algorithm to determine if a graph is *two-colorable*, that is if there is a coloring of the graph that uses no more than two colors. You will also determine how many connected components there are. **Please do not use any built-in or third party graph functions/libraries. Implement your own.**

The input graph will be undirected and simple (no self loops or multiple edges). I provide a sample input graph on PolyLearn. You will certainly need more test cases, create these yourself.

The format will be a simple edge list. There will be n vertices and m edges. Each edge in the graph will be represented by a pair of vertex identifiers. Note that I will not give you graphs with isolated vertices. All edge lists will begin at 0.

For example the following graph is represented by the list:



```
1 2
1 3
2 3
1 4
0 5
4 3
3 0
```

What to bring with you to lab:

- Your pseudocode printed or handwritten.
- A list of at least three interesting test cases for your algorithm. Draw the graphs and be able to explain why these are important tests.

What to turn in on PolyLearn:

- Your algorithm should take a graph and state whether it is two-colorable or not and state the number of connected components.
  - For example: Your graph would take the edge list above and return that it is not two-colorable and has 1 connected component.
- Submit all of your source code along with a shell script driver named "asgn2.sh". For record-keeping purposes, submit your source files, not precompiled class or object files. I should be able to run your program in a Unix environment by typing, "./asgn2.sh <input file name>". Your program must read

input from a file whose name is provided as a command line argument and print output to stdout. Your output will be tested using diff, so it must match exactly. Please don't zip your files.

You will be provided with sample input and output. An example submission for a fake programming assignment can be found on PolyLearn.