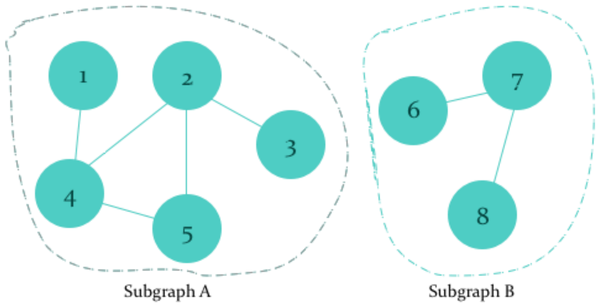
CPE 202

Assignment 5: Finding connected components and testing if a graph is bipartite

Finding connected components and graph coloring are important problems in computer science.

Knowing the **connected components** in an undirected graph provides information about sets of vertices such that each pair of vertices has a path between them. It is a list of the sub-graphs within the larger graph.

For the graph to the right, we have two sub-graphs within the larger graph.



A **coloring** is an assignment of a color to each vertex in the graph so that no two adjacent vertices have the same color. Colorings are important in modeling problems where a set of objects must satisfy certain constraints. In this assignment, we will determine if a graph is bicolorable (also referred to as bipartite). If a graph is bicolorable, it means that each vertex can be assigned one of two colors such that no two adjacent vertices (i.e. two vertices with an edge connecting them) have the same color.

More information on bicoloring and bipartite graphs can be found here: https://en.wikipedia.org/wiki/Bipartite_graph .

For this assignment, you will implement algorithms based on Breadth and Depth First Searches to:

1.) Determine the connected components of an undirected graph, using **Depth First Search**.

2.) Determine if the graph is bipartite (bicolorable), using **Breadth First Search**.

As noted above, a graph is bipartite or bicolorable if the vertices of the graph can be assigned labels (e.g. red and black), such that no two adjacent vertices have the same label.

You may assume that the graph is **undirected and does not contain self-loops (**edges from a vertex to itself).

You may want additional functions in your Graph and Vertex classes in order to implement the requirements, but you must implement the functions as specified in the starter file.

SUBMISSION

Three files:

- **graph.py** containing a classes **Vertex** and **Graph**
- **graph_tests.py** – tests for your Vertex and Graph methods
- **queue_array.py** – implementation from Lab 3, used for BFS