Fall 2018 CS313 Project

Implementation of a GUI to work with Graphs.

Reminder: This is a pass/fail assignment. If you fail the assignment you will lose half a grade from your course grade. If you pass the assignment and complete a fully working version of the project you will gain a half grade adjustment to your course grade,

You do not have to do a perfect job to pass the assignment, but you must submit something that compiles and creates a GUI that performs most of the required tasks. The surest way to fail this assignment is to use code written by somebody else (in or outside of the class) or to share code with another student. I will use an automated system that detects similarity between different pieces of code.

The final deadline for this project is Wednesday 12/12/2018. However there are preliminary milestones that you should try to achieve. You should submit preliminary versions of the project to check that you have achieved them. The deadlines for these milestones are as follows:

- Phase 1: Due 11/21/18: Code to show a Gui screen with required buttons. The help button should work. Also pressing any button should release all other radio buttons.
- Phase 2: Due 11/28/18: Additional code to add, move and remove vertices in response to user mouse clicks on the screen.
- Phase 3: Due 12/05/18: Additional code to add and remove edges and display graphs on the screen.
- Phase 4: Due 12/12/18: Complete code which will now show connected components of a graph in different colors and highlight any cut vertices. This is the version that will receive a grade.

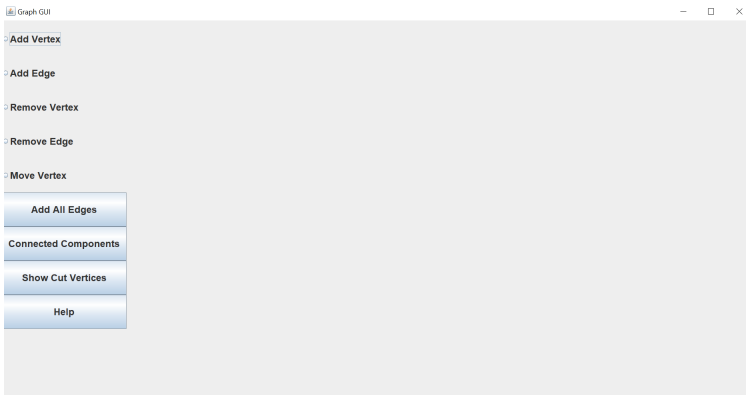
I advise you to try to submit work for the preliminary phases. This is optional, but it is a way for me to give you feedback about whether you are on track with the project. If your project does not complete the first three phases it will not pass.

When you submit any version of your project (preliminary or final) you should submit only one file named GGxxxx.java where xxxx is changed to be the last 4 digits of your 8 digit ID number. For example, if your id number was 31415926 you would submit a file called GG5926.java.

The main class in this file must be a public class GG5926 but no other classes that you use can be public (because only one public class is allowed in a file). Although this is not an ideal way to organize code it simplifies my task in grading. I will not accept any other type of submission either for the final version or the earlier milestones. Submit the homework by emailing your one file as an attachment from your own Queens College email account to Alexander.Ryba@qc.cuny.edu. The subject for your email must be: CS313 Project Phase X, where X is the phase corresponding to your file. Late work will

not be considered for any phase of the project. Email from addresses outside Queens College will not be accepted.

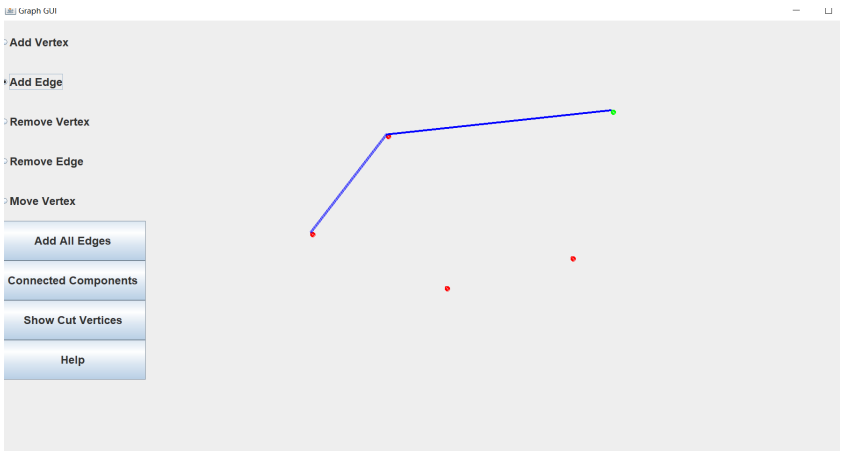
The gui that you make should offer the features shown here. This screenshot has been made before any graph has been entered. Choosing the *Help* button should pop up another window with instructions about how to run your gui.



In order to pick vertices, the user selects the radio button marked *Add Vertex* and selects positions of vertices in the right half of the gui by clicking the mouse. Each mouse click generates a vertex of the graph and these vertices are marked in red as they are added.

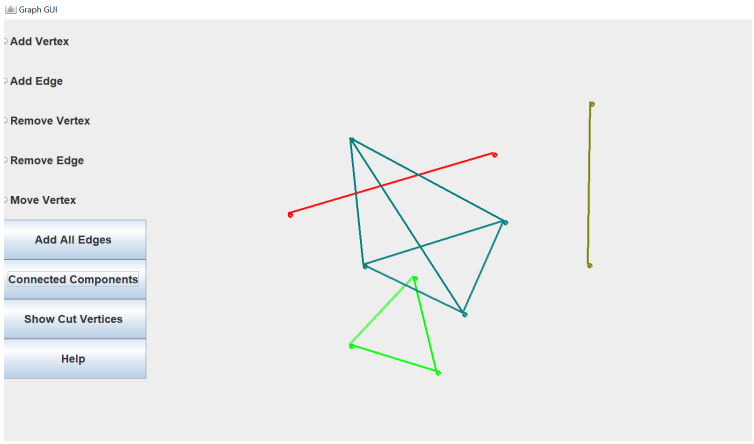
The radio button *Remove Vertex* allows the user to remove a vertex by clicking on it. Later when there are edges attached to a vertex, removing it should also remove these edges. The radio button *Move Vertex* should allow the user to select a vertex by clicking on it and then click at a new location to move the vertex. Of course any edges attached to the vertex must move with it. You need to make the gui accept mouse clicks that are reasonably close to a vertex since it is impossible for a user to click its exact pixel location.

The user selects the radio button marked *Add Edge* to add edges. An edge is made by clicking close to the two vertices that specify its ends. After the first end vertex of an edge is selected it is highlighted in green until the second end has been selected too at which time the edge is added to the graph and drawn on the gui in blue.

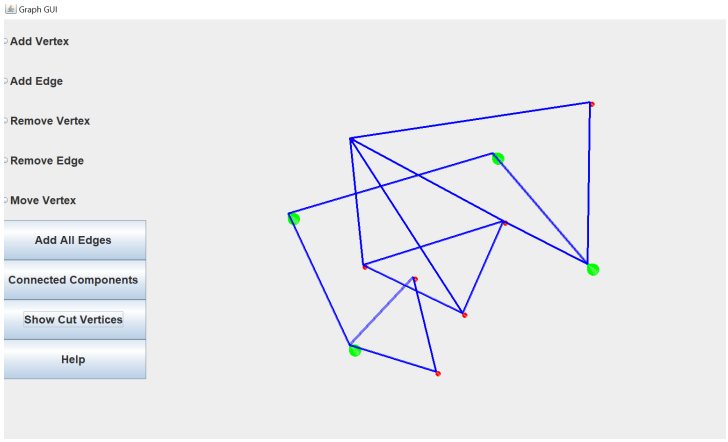


The user selects the radio button marked *Remove Edge* to remove edges. Once the button is selected, any click on or near an edge removes it (but leaves the vertices at its ends). The button marked *Add All Edges* is a shortcut that add in all possible edges between pairs of vertices.

The button *Connected Components* makes the gui show the different components of the graph in different colors. An example of how it should operate is shown here:



The button *Show Cut Vertices* makes the gui highlight all cut vertices of the graph. An example of how it should operate is shown here:



Although the first steps of this project just involve Gui code, later steps will require work with a Graph data structure. This is the topic of Chapter 14 of the text which you should read. You should also refer to the textbook for CS220 to look up standard terminology about graphs.

You should probably plan to use a number of classes in your project. In addition to your main class GGxxxx which has the job of displaying the gui on screen, you might consider having a class Vertex, a class Edge, a class Graph, a class for the Help screen, listener classes for your Gui and a class that represents

the panel of the Gui in which the graph is displayed.

The following code fragment could help you get started with an implementation that uses swing classes.

```
public class GGxxxx extends JFrame {

   GraphPicturePanel picture;  // you would have to make this class
                               // to implement behavior of the picture

   public static void main(String args[]) {
      try {
         UIManager.setLookAndFeel(UIManager
               .getCrossPlatformLookAndFeelClassName());
      } catch (ClassNotFoundException | InstantiationException
            | IllegalAccessException | UnsupportedLookAndFeelException e) {
      }
      new GraphGui();
   }

   public GraphGui() {
      super("Graph GUI");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // code to layout gui components omitted
   }

   public void paint(Graphics g) {   // method to be sure the
                                     // picture gets redrawn as it is modified
   // if you want to use this plan your GraphPicturePanel
   // needs a method with title line:
   //    public void paintComponent(Graphics g) {
      super.paint(g);
      picture.repaint();
      setVisible(true);
   }

   // other gui code omitted
}
```

You will also need a class that implements an `ActionListener` to respond to user selections of buttons and a class that extends `MouseAdapter` to detect mouse clicks in the picture panel.

I will compile and run your program on venus. You should make sure that if you move your file GGxxxx.java to venus you can compile it with the command:

javac GGxxxx.java

If you use unusual packages and libraries this might be a problem. If you cannot find a work around for this you should come to discuss your project and

how you have implemented it with me. Do not wait until the final stage to do this.

Running your GUI on venus can be a little tricky because you need to find a way to get it to show up on a local screen, but you should still be able to compile on venus.

Some common issues that could prevent Phase 1 work from compiling on venus are:

1. An initial package line (inserted by eclipse) was left as the first line of the file. This should be removed because it refers to a particular eclipse set up and directory structure that will not be present on venus or other machines.
2. The file contains no public class.
3. The public class contains no main method.
4. The public class has a different name from the file.

Also make sure to set a default close operation so that you program ends when the GUI is closed.