Last name _____

First name _____

LARSON—MATH 356—LAB WORKSHEET 04
Our First Graph Algorithm!

1. Log in to your Sage/CoCalc account.
   (a) Start the Chrome browser.
   (b) Go to `http://cocalc.com` and sign in.
   (c) You should see an existing Project for our class. Click on that.
   (d) Click "New", call it **s04**, then click "Sage Worksheet".
   (e) For each problem number, label it in the Sage cell where the work is. So for Problem 1, the first line of the cell should be `#Problem 1`.
   (f) When you are finished with the worksheet, click "make pdf", email me the pdf (at `clarson@vcu.edu`, with a header that says **Math 356 s04 worksheet attached**).

Graph-6 Strings

2. Find the order and size of the tetrahedron graph. Use `tetra=graphs.TetrahedralGraph()`.

3. What is a representation of a graph you can easily send by email—and reconstruct? (How can you avoid ever coding the tetrahedron graph ever again?)

Our First Graph Algorithm

4. Let's write a function `is_complete(G)` that takes a graph G and outputs `True` if G is a complete graph and `False` if it is not. What should our function do?

Random Graphs

5. One way to make a graph is to start with a number of vertices and then for each pair of vertices n and m, flip a coin to decide whether to put an edge between those vertices. `random()` gives a random number between 0 and 1. Try this:

```
G=Graph(10)
for i in [0..9]:
    for j in [0..9]:
        if i<j and random()<.5:
            G.add_edge(i,j)
G.size()
G.show()
```

6. The study of *random graphs* is huge and important and was initiated in a 1959 paper of Erdős and Renyi. Sage has a built in function to do this: `graphs.RandomGNP(n,p)`, where n is the number of vertices you want, and p is the probability of an edge ($0 \leq p \leq 1$). To simulate a coin flip, use $p = .5$. Run the following code a few times.

```
G=graphs.RandomGNP(10,.5)
G.size()
G.show()
```

Testing if a Graph is Connected

7. Write a function `neighbors(G,v)` that takes a graph G and a vertex v as input and returns the vertices that are adjacent to vertex v (that are *neighbors*).

8. Write a function `neighbors(G,S)` that takes a graph G and a set S of vertices as input and returns the vertices that are adjacent to *any* vertex v in S.

9. Write a function `new_neighbors(G,S,T)` that takes a graph G and a sets S and T of vertices as input and returns the vertices that are adjacent to *any* vertex v in S but are not already in T.

10. Write a function `is_connected(G)` that takes a graph G as input and returns TRUE is G is connected and FALSE if it is not connected.

Testing if a (Connected) Graph is Bipartite

11. Write a function `is_bipartite(G)` that takes a connected graph G as input and returns TRUE is G is bipartite and FALSE if it is not bipartite.

Concepts from Our Text

These include: *size, order, complete graphs, bipartite graphs, isomorphic graphs, subgraph, complement, incidence matrix, adjacency matrix, degrees, minimum degree, maximum degree, identical graphs.*

These are all built-in to Sage/Cocalc!

12. How can we test is a graph is complete in Sage?

13. How can we test is a graph is bipartite in Sage?

14. How can we test if two graphs are isomorphic in Sage?

15. How can we find the complement of a graph in Sage?

16. How can we produce an induced subgraph with a particular subset $V' \subseteq V$?

17. How can we find the vertices of a given graph G?

18. How can we find the edges of a given graph G?

19. How can we test if two graphs are identical?

20. How can we find the degrees of a graph?

21. How can we find the maximum degree?

22. How can we find the minimum degree?