

**LARSON—MATH 255—CLASSROOM WORKSHEET 33**  
**Graph Theory.**

1. (a) Start the Chrome browser.  
(b) Go to `http://cocalc.com`  
(c) Login using **your VCU email address** .  
(d) Click on our class Project.  
(e) Click “New”, then “Worksheets”, then call it **c33**.  
(f) For each problem number, label it in the Sage cell where the work is. So for Problem 2, the first line of the cell should be **#Problem 2**.

### Graphs & Graph Theory

A **graph** is a mathematical object consisting of *dots* and *lines* (also called *vertices* and *edges*). The *order* of a graph is the number of vertices it has. The *size* of a graph is the number of edges it has. We can create our own graph using the `Graph()` constructor, and the `add_vertex()` and `add_edge()` methods.

2. Make the following graph, called the *bull*. Its built-in to Sage. Evaluate: `bull = graphs.BullGraph()`. Then view it: `bull.show()`.

(**Adjacency Matrices**) Another way to represent a graph with order  $n$  is with an  $n \times n$  *adjacency matrix*  $A$ . If the vertices of the graph are  $\{v_0, v_1, \dots, v_{n-1}\}$  (or  $\{0, 1, \dots, n-1\}$  for short) then the  $A_{i,j}$  is 1 if there is an edge from vertex  $i$  to vertex  $j$ , and 0 if there is not.

3. Try:

```
bull.adjacency_matrix()
```

Make sure you understand the pattern of 0's and 1's.

4. Let's get acquainted with paths, cycles, stars, and complete graphs. Try:

```
@interact
def i_graph(graph=selector(["path", "cycle", "star", "complete"],
    label="Select a graph", default="path", order=slider(3,20,1,3))):

    dict={"path":graphs.PathGraph(order),
        "cycle":graphs.CycleGraph(order),
        "star":graphs.StarGraph(order),
        "complete":graphs.CompleteGraph(order)}
    g=dict[graph]
    order = g.order()
    size=g.size()
    print("This graph has {} vertices and {} edges".format(graph,order,size))
    g.show()
```

5. (**Random Graphs**) 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. Try this:

```
g=Graph(10)
for i in [0..9]:
    for j in [0..9]:
        if i<j and random() < 0.5:
            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, 0.5)
g.size()
g.show()
```

7. (**Challenge**) How many edges does a random graph with 10 vertices have *on average*? How can we investigate this?

### Getting your classwork recorded

When you are done, before you leave class...

- Click the “Make pdf” (Adobe symbol) icon and make a pdf of this worksheet. (If CoCalc hangs, click the printer icon, then “Open”, then print or make a pdf using your browser).
- Send me an email with an informative header like “Math 255 - c33 worksheet attached” (so that it will be properly recorded).
- Remember to attach today’s classroom worksheet!