Recursion—Graph Theory

1. (a) Start Chrome browser.

(b) Go to http://cocalc.com and “Sign In”.

(c) Click project Math 356.

(d) Click “New”, call it c11, then click “Sage Worksheet”.

A recursive function is a function that calls itself. It must always have a base case so that the recursion eventually stops.

2. Here is an example of a recursive definition of the factorial function. The base case here is the case where the input is 0 or 1.

```python
def factorial(n):
    if n==0 or n==1:
        return 1
    else:
        return n*factorial(n-1)
```

Now try factorial(0), factorial(1), factorial(2), factorial(3), and factorial(10).

Sage includes the graphs class which contains a number of methods. Some of these include constructors for making well-known graphs. “pete” is an arbitrarily chosen name in what follows. We could use any other name instead!

3. Evaluate:

```python
pete=graphs.PetersenGraph()
pete.show()
```

The order of a graph is the number of vertices it has. The size of a graph is the number of edges it has. How many vertices and edges does the Petersen graph have? Evaluate Pete.order() and Pete.size().

4. Find the order and size of the icosahedron graph. Use g2=graphs.IcosahedralGraph()

5. Find the order and size of the dodecahedron graph. Use g3=graphs.DodecahedralGraph()
6. Find the order and size of the tetrahedron graph. Use `g4=graphs.TetrahedralGraph()`.

7. Find the order and size of the octahedral graph. Use `g5=graphs.OctahedralGraph()`.

8. Let's get acquainted with paths, cycles, stars, and complete graphs. Evaluate the following Sage `Interact` and play with the features you see:

   ```python
   @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]
print "This graph has %s vertices and %s edges" %(g.order(),g.size())
g.show()
```

9. We can create our own graph using the `Graph()` constructor, and the `add_vertex()` and `add_edge()` methods. Let's make a cycle on 5 vertices. First initialize the graph and make the vertices:

   ```python
g=Graph()
for i in [1..5]:
    g.add_vertex()
g.show()
```

Notice that the vertex labels start at 0. Now make the edges:

   ```python
for i in [0..3]:
    g.add_edge(i,i+1)
g.show()
```

You're still missing an edge. What will you add to the code to get the missing edge?

10. Try the code from pp.17-18 of our book.

11. What words (terms, concepts) don’t you know yet? Make a list.

12. Google one of these terms. Try `G.your_term?` to see if you can get information from Sage. Evaluate the value of this term for `pete`, `g2`, `g3`, `g4`, `g35`.

13. Try and make a general `conjecture` about your term for paths, cycles, or complete graphs.