LARSON—MATH 255–CLASSROOM WORKSHEET 23
Problems & Graphs

1. (a) Start the Chrome browser.
   (b) Go to `http://cocalc.com`
   (c) You should see an existing Project for our class. Click on that.
   (d) Click "New", then "Sage Worksheet", then call it **c23**.
   (e) For each problem number, label it in theSAGE cell where the work is. So for Problem 1, the first line of the cell should be `#Problem 1`.

Problems

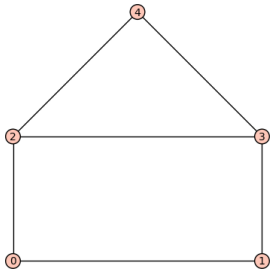
2. (**Ramanujan**) 2, 9, 16, etc. can be written (uniquely) as the sum of 2 cubes (1^3+1^3, 1^3+2^3, 2^3+2^3, etc.). Find the smallest integer which can be written as the sum of 2 cubes in 2 different ways.

3. (**Open ended**) Find all real values of a so that the sequence $\{a_n\}_{n\geq 0}$ defined by $a_0 = a$ and $a_{n+1} = a_n^2 - 2$ for $n \geq 0$, converges.

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.

4. Make the following graph, called "the house". Start by letting `house=Graph(5)`. When you are done you can view it with `house.show()`.



5. Try the following Sage Interact which shows some famous graphs and uses a *dictionary*:

```
@interact
def i_graph(graph=selector(["icosahedron", "dodecahedron",
    "tetrahedron", "octahedron"],
    label="Select a graph", default="tetrahedron")):

    dict={"icosahedron":graphs.IcosahedralGraph(),
    "dodecahedron":graphs.DodecahedralGraph(),
    "tetrahedron":graphs.TetrahedralGraph(),
    "octahedron":graphs.OctahedralGraph()}
    g=dict[graph]
    order = g.order()
    size=g.size()
    print("The {} has {} vertices and {} edges".format(graph,order,size))
    g.show()
```

6. 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()
```

   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 $\{v_0, v_1, \ldots, v_{n-1}\}$ (or {0,2,...,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.

7. Try:

```
house.show()
house.adjacency_matrix()
```

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

Getting your classwork recorded

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

1. 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).

2. Send me an email with an informative header like "Math 255 - c23 worksheet attached" (so that it will be properly recorded).

3. Remember to attach today's classroom worksheet!