LARSON—MATH 255–CLASSROOM WORKSHEET 25
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 **c25**.
   (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. (**Dynamical Systems**) 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.

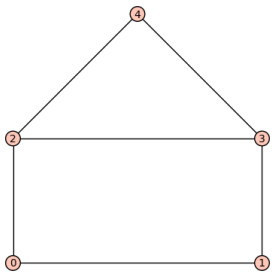
3. (**Airplane Seats**) One hundred people have assigned seats on a 100-seat airplane. The first person that gets on takes a random seat. The second person that gets on takes her seat if it is unoccupied. Otherwise the person takes a random seat. The third person that gets on takes his seat if it is unoccupied. Otherwise the person takes a random seat. And so on. What is the probability that the last person ends up in his or her assigned seat?

4. (**Estimating** π) We can randomly place points in a 1-by-1 square. We also know the equation of a circle centered in this square. How can we use those ideas to estimate a value for π?!?!

Graphs & Graph Theory

A **graph** is a mathematical object consisting of *dots* and *lines* (also called *vertices* and *edges*).

5. The following graph is called "the house". Start by letting `house=graphs.HouseGraph()`. When you are done you can view it with `house.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.

6. Try `house.adjacency_matrix()`.Make sure you understand the pattern of 0's and 1's.

   Now we will define a new graph function for computing the hard-to-compute *stability number* of a graph. The *stability number* of a graph is the largest number of vertices in the graph that have no edges between them. So if `g=graphs.PetersenGraph()`, its stability number is 4.

7. Find the stability number for `k5=graphs.CompleteGraph(5)`. Use `k5.show()` to help you visualize.

8. We'll want to look at each possible pair of vertices in a set S of vertices to see if there's an edge between them. One way to do that is see if a given pair is in the list of edges. We can get the edges of a graph g using `g.edges(labels=False)`. Try it for the Petersen graph and $k5$.

   Now we will write an algorithm to find the stability number.

9. First let's write a test for whether or not a specific set S of vertices is stable. Then one possible algorithm involves testing every subset of vertices. If S is stable then the test for (i, j) will not be an edge (will not be in the list of edges) of graph g for each possible pair i, j of vertices.

```
def is_stable(g, S):
    E=g.edges(labels=False)
    for i in S:
        for j in S:
            if (i,j) in E:
                return False
    return True
```

10. Test this for some (sub)sets of vertices of the Petersen graph.

Subsets and Counting

The "naive" (and inefficient) way to find a largest stable set in a graph is to test every subset of vertices, check if it is stable, and then keep track of the largest one you've seen up to that point. Let's see why its inefficient.

11. Suppose you have a very small graph, with just 3 vertices. How many subsets of the vertex set would you have to look at? Run: `Subsets([0,1,2])`.

12. How many is that? We can turn the *Subset generator* into a list and use the Python list counting function *len*. Run *len(Subsets([0,1,2]))*. (Don't ever try this on a set that's much larger—the list gets created in memory and might freeze your computer as it runs out of RAM).

13. How many subsets will we create for a graph that has n vertices? For each vertex v we will generate every possible set containing v and every possible set not containing v. So that will be 2 possibilities for each of the n vertices. That's 2^n subsets. (For our earlier example, thats $2^3 = 8$ subsets).

14. (**Computational Aside**) A graph with 100 vertices is considered "small" (we'd like to find information about graphs with millions of vertices). How long (minutes, hours, days, years?) would it take for an algorithm to perform 2^{100} computational steps. Assume you can perform one million (10^6) steps per second.

15. (**The Algorithm Implemented**). Write a function `stability_number(g)` that takes a graph g as input, forms every possible subset of vertices, checks if that set of vertices is stable, and updates the largest stable set found so far, and returns the number of vertices in a largest stable set.

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 - c25 worksheet attached" (so that it will be properly recorded).
3. Remember to attach today's classroom worksheet!