LARSON—MATH 255–CLASSROOM WORKSHEET 36
Graph Theory & Computationally Hard Problems.

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 **c36**.
   (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*). 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.

2. (**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.

3. Here is 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.

```
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

def naive_maximum_stable_set(g):
    stable = []
    L=subsets(g.vertices())
    for S in L:
        if is_stable(g,S)==True:
            if len(S) > len(stable):
                stable = S
    return stable

def stability_number(g):
     return len(naive_maximum_stable_set(g))
```

Now we will use this function to find a maximum stable set of every graph you previously hand-calculated. Can you *prove* the set is maximum?

4. Use this function to find a maximum stable set of $p2$. Check that it agrees with your hand calculation.

5. Use this function to find a maximum stable set of $p3$. Check that it agrees with your hand calculation.

6. Use this function to find a maximum stable set of $p4$. Check that it agrees with your hand calculation.

7. Use this function to find a maximum stable set of $p5$. Check that it agrees with your hand calculation.

8. Use this function to find a maximum stable set of $c5$. Check that it agrees with your hand calculation.

9. Use this function to find a maximum stable set of the Petersen graph.

10. When you are testing a function you might want to print more information in order to see if its doing what you expect. Have your `stability_number` print the maximum stable set it found, and `show` your graph. Then you can see for your self if the set is stable. Try it with graphs that you've named.

What else can be done?

The next big idea for finding a maximum stable set was due to Tarjan and Trojanowski in the 1970s: they noted that each vertex v of a graph is either in a maximum stable set *or* it is not. And, if v is in a maximum stable set then none of the vertices it is touching (that it is *adjacent* to is), called the *neighbors* of v, can be in that set.

11. Find the neighbors of vertex 9 in the Petersen graph.

12. How can the Tarjan-Trojanowski idea be turned in to a *recursive* algorithm?

Getting your classwork recorded

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

(a) 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).

(b) Send me an email with an informative header like "Math 255 - c36 worksheet attached" (so that it will be properly recorded).

(c) Remember to attach today's classroom worksheet!