LARSON—INFO 790–CLASSROOM WORKSHEET 02
Using CONJECTURING on CoCalc

1. Log in to CoCalc.

   (a) Start the Chrome browser.

   (b) Go to `https://cocalc.com`

   (c) Login (**your VCU email address** is probably your username).

   (d) You should see an existing Project for our class. Click on that.

   (e) Click "New", then type **790-c02** into the box, and click "Jupyter Notebook".

   (f) When your notebook opens look on the upper-right to make sure the SageMath kernel is running.

Conjecturing Invariant Bounds

We can use the `conjecturing` program to conjecture upper and lower bounds for an *invariant* of an mathematical object (number, matrix, graph, etc). An *invariant* in this context means any number associated with that object. So, for instance, the determinant of a matrix is a matrix-invariant.

Inequalities show up everywhere in mathematics; famous ones include the Cauchy-Schwartz inequality. Investigating bounds can be of enormous practical importance: bounds are useful when we want to reduce a *search space* where the answer to some question may be (for instance optimizing a discrete function).

2. Load "conjecturing.py" by running `load("conjecturing.py")`. (Your "790-c02.ipynb" must be in your Home/root directory for this to work, and there should be an "expressions" and "conjecturing.py" file there too).

3. Try this first simple example. Interpret the conjectures. Are they true?

```
objects = [2,3,4]
invariants = [Integer.nbits, Integer.ndigits, Integer.sqrt]
invariant_of_interest  = invariants.index(Integer.nbits)
conjecture(objects, invariants, invariant_of_interest, upperBound = True)
```

How does Conjecturing work?

   (a) What is the *Truth* heuristic?

   (b) What is the *Significance* heuristic?

(c) Re-run the code to see some under-the-hood details using the *verbose* and *debug* options:

```
objects = [2,3,4]
invariants = [Integer.nbits, Integer.ndigits, Integer.sqrt]
invariant_of_interest  = invariants.index(Integer.nbits)
conjecture(objects, invariants, invariant_of_interest, upperBound=True,
     verbose=True, debug=True)
```

Conjecturing Properties

4. Try this first simple example. Interpret the conjectures. Are they true?

```
objects = [2,3,4]
properties = [Integer.is_prime, Integer.is_square, Integer.is_squarefree]
property_of_interest  = properties.index(Integer.is_prime)
propertyBasedConjecture(objects, properties, property_of_interest,
     sufficient=True)
```

5. How does the *Truth* heuristic work in the "properties" case?

6. How does the *Significance* heuristic work in the "properties" case?

7. Re-run the code to see some under-the-hood details using the *verbose* and *debug* options:

```
objects = [2,3,4]
properties = [Integer.is_prime, Integer.is_square, Integer.is_squarefree]
property_of_interest  = properties.index(Integer.is_prime)
propertyBasedConjecture(objects, properties, property_of_interest,
    sufficient=True, debug=True, verbose=True)
```