* The unique games conjecture says that approximating the value of the unique game is NP-Hard.
  + The value is the fraction of constraints that can be satisfied by any assignment.
* Because many problems can be “cast” to other problems, this conjecture confers that many problems are NP-Hard to approximate.
* It is much easier to find the value of the unique game for satisfiable instances but finding the value of unsatisfiable instances is much more difficult even to approximate.
* A two-prover one-round game is an instance of a unique game if for every question and every answer by the first player, there is exactly one answer from the second player that results in a win for the players, and vice versa. The value of the game is the maximum winning probability for the players over all strategies.
* Verifying graph coloring is O(n^2)
  + e = n(n-1)/2 for a complete graph, and you would need to check every edge for same color neighbors

The interesting thing about the unique games conjecture to me is that solving for the value of a satisfiable problem is (relatively) simple. Once you find a satisfied state of the problem, the answer is 1. However, trying to find the value of an un-satisfiable problem is much more complicated because (in theory) the only way to do it is to try every single possible state and calculate the value to find the best.

Propose:

Implement the “random game” that randomly creates a very simple graph (5-6 nodes with random edges) and applies 2-3 constraints to each edge. Run an algorithm that exhaustively searches every single possibility of the graph to calculate the value.

Run this “random game” ten thousand times, logging the results of each game. Specifically, we need to record things like the best value, average value, count of best values, median of values, range of values. This data should demonstrate that sometimes, the count of best values is only 1, which means an approximation algorithm won’t suffice if you are looking for the perfect answer. In other cases, the count of best values might be a very large number meaning an approximation algorithm would likely give a correct result.

Constraints:

* Use a maximum of 4 colors, red green blue yellow

Constraints on Edges:

* Nodes must be X color.
* Nodes must not be X color.
* Nodes must be different colors.
* Nodes must be the same color.
* (Can also try simply applying constraints such as “edge must be green to blue” or “blue to yellow”)