Team Members:

Matthew Zheng, PID: 730092528

For both algorithms, the Constraint Satisfaction Problem is represented in the *csp* class. This class contains three instance variables:

- Variables represents the variables in the CSP. In this case, they are the states in the map.
- Domains represents values taken on by the variables in the CSP. It is used slightly different in backtracking search versus local search. In backtracking search, it is used to represent all possible values that the variable could take on as dictated by Arc Consistency 3. In local search it is used to represent the current value taken on by the variable, modified during the process of hill-climbing.
- Constraints represents all the arcs, aka the constraints on any given variable.

Backtracking Search:

My implementation utilizes Arc Consistency 3 in the Backtracking Search. The backtracking search runs recursively on the next adjacent unassigned variable, or if none exists any unassigned variable in the CSP. After each attempted assignment (that has no immediate conflicts), Arc Consistency 3 is used to propagate changes to the rest of the CSP. Should AC3 find that it impossible to be consistent, the value is discarded and the search recurses up. Otherwise, the process continues until every variable is assigned.

AC3 operated by creating a worklist of all arcs for each variable in the CSP. Then, attempting each value in the domain of each variable, it ensures that each arc in the CSP is consistent by running arc_reduce on each arc which ensures that there are values within the domains of both ends of the arc that are legal.

Local Search:

My implementation performs random-restart hill climbing. At first, colors are randomly assigned with no regards to constraints. This coloring is kept as a base to restart at. Then, a random variable is chosen to hill climb from. The hill climb checks all possible values for the chosen variable and selects the one that produces the minimal number of conflicts. If no such value, the hill climb function reports a failure and the local search will restart. If the total number of conflicts decreases, the next variable chosen is the most constrained variable and hill climbing continues. This process repeats until either a solution is found (total conflicts is 0) or the hill climb fails in which case the local search restarts from the base.