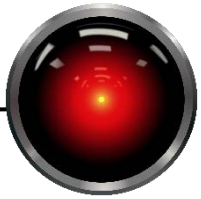


Assignment 4

Part I: Questions (20 points each)

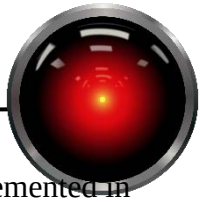
1. Write a constraint satisfaction problem specification for a 4 color map problem with colors: pink, green, blue, and red as per the following map:

2. Given the vehicle assembly example given in slides 7-9, draw the constraint graph.
3. Explain in your own words how conflict-directed backjumping works. As always, be very careful to avoid plagiarizing.

Part II: Programming Assignment – 120 points

The code package for this assignment on Blackboard contains a partially implemented constraint satisfaction problem class for solving Sudoku puzzles. The sudoku module implements the class Sudoku that is derived from a CSP class (both in module csp_lib). The code and comments explain the data structures that are used and contain a sample instantiation of a Sudoku puzzle. The module also provides two sample puzzles, one that can be solved with constraint satisfaction, the other without.

There is no work to do in the sudoku and csp modules other than to read them and understand how they work. The task in this assignment is to implement AC3 constraint propagation, backtracking search, and a driver. Template functions are provided for AC3 and backtracking_search in modules constraint_prop and backtrack. Both of these operate on a sudoku puzzle and update the current assignments. When calling your



backtrack search, use the minimum remaining values heuristic (which is implemented in csp_lib.backtrack_util).

Your driver program should create both the easy and hard sudoku problems and then solve them.

To turn in:

Submit driver.py, backtrack.py, constraint_prop.py and any other routines that you create. As always, turn in a print out and electronic versions. Your print out should show the initial sudoku states for each puzzle as well as the result of AC3 constraint propagation and if needed, backtrack search.