COSC 76, Fall 2020, PA-4, Edmund Aduse Poku

Q1b.:

a) Description

GENERIC CSP CLASS AND SOLUTION CLASS

I implemented a generic CSP class that has 4 instance attributes: variables, domains, constraints and neighbors. It also has 3 methods: arc_is_consistent (determines if a given arc is consistent or satifies the problem constraints), ass_is_consistent (checks to see if, after an assignment, the assignment is consistent). The solution class takes the solution to a given csp problem and represents it in succint, human readable form using the representation methods described in the corresponding specific CSP class.

BACKTRACKING SEARCH, HEURISTIC AND INFERENCES FUNCTIONS

I implemented the backtracking search method in its own file called search.py. The backtracking search method has the ability to switch between using none, some or all of the heuristic functions: "mrv", "lcv", and degree_heuristics. The heuristic functions have been inmplemented in their own file called heuristics.py. It also has the ability to switch between none or one of the implemented inferences functions: forward_checking and "mac". All the heuristic and inferences methods function as described in the textbook.

MAP COLORING PROBLEM

I extended the generic CSP class to implement a specific class for the Map Coloring Problem. The class has a convert_to_csp method that converts the human readable Map Coloring Problem into a generic csp that can be solved by the CSP class. The class convert_to_readable also has a method that reverts the solution into a human readable form.

CIRCUIT LAYOUT PROBLEM

Similar to what I did for the Map Coloring Problem, I extended the generic CSP class defined in the csp.py file and added methods to convert a given circuit problem to a generic csp. There is also a method to convert the solution back to a human readable form (circuit).

b) Yes, all the algorithms implemented work (perfectly). Running the test cases for the Map Coloring Problem and the Circuit Board Problem gives the following results, just as expected:

MAP COLORING PROBLEM

MapColoringCSP solved by backtracking in runtime: 0.000053 seconds using the following: no heuristics and no inference the solution suggests that the colors be assigned in the manner: {'WA': 'g', 'NT': 'b', 'Q': 'g', 'NSW': 'b', 'SA': 'r', 'V': 'g', 'T': 'g'}

MapColoringCSP solved by backtracking in runtime: 0.000742 seconds using the following: 'MRV' heuristics and no inference the solution suggests that the colors be assigned in the manner: {'WA': 'g', 'NT': 'b', 'Q': 'g', 'NSW': 'b', 'SA': 'r', 'V': 'g', 'T': 'g'}

MapColoringCSP solved by backtracking in runtime: 0.000687 seconds using the following: 'MRV', 'LCV' heuristics and no inference the solution suggests that the colors be assigned in the manner: {'WA': 'g', 'NT': 'b', 'Q': 'g', 'NSW': 'b', 'SA': 'r', 'V': 'g', 'T': 'g'}

MapColoringCSP solved by backtracking in runtime: 0.000159 seconds using the following: degree_heuristics, 'MRV', 'LCV' heuristics and no inference the solution suggests that the colors be assigned in the manner: {'SA': 'g', 'WA': 'b', 'NT': 'r', 'Q': 'b', 'NSW': 'r', 'V': 'b', 'T': 'g'}

MapColoringCSP solved by backtracking in runtime: 0.000202 seconds using the following: degree_heuristics, 'MRV', 'LCV' heuristics and 'forward checking' inference the solution suggests that the colors be assigned in the manner: {'SA': 'g', 'WA': 'b', 'NT': 'r', 'Q': 'b', 'NSW': 'r', 'V': 'b', 'T': 'g'}

MapColoringCSP solved by backtracking in runtime: 0.000056 seconds using the following: degree_heuristics, 'MRV', 'LCV' heuristics and 'mac' inference the solution suggests that the colors be assigned in the manner: {'SA': 'g', 'NT': 'r', 'NSW': 'r', 'WA': 'b', 'Q': 'b', 'V': 'b', 'T': 'g'}

CIRCUIT LAYOUT PROBLEM

CircuitBoardCSP solved by backtracking in runtime: 0.000104 seconds using the following: no heuristics and no inference the components should be laid out as:

bbbbbb. .... aa.gg.. aa.gg.. aa..... aa.eeee aa.eeee

using the coordinates: {'a': (0, 0), 'b': (0, 6), 'e': (3, 0), 'g': (3, 3)}

CircuitBoardCSP solved by backtracking in runtime: 0.000094 seconds using the following: 'MRV' heuristics and no inference the components should be laid out as:

bbbbbb. .... aa.gg.. aa.gg.. aa..... aa.eeee aa.eeee

using the coordinates: {'a': (0, 0), 'b': (0, 6), 'e': (3, 0), 'g': (3, 3)}

CircuitBoardCSP solved by backtracking in runtime: 0.000389 seconds using the following: 'MRV', 'LCV' heuristics and 'mac' inference the components should be laid out as:

bbbbbb. .... aa.eeee aa.eeee aa..... aa..gg. aa..gg.

using the coordinates: {'a': (0, 0), 'b': (0, 6), 'e': (3, 3), 'g': (4, 0)}

CircuitBoardCSP solved by backtracking in runtime: 0.000222 seconds using the following: degree_heuristics, 'MRV', 'LCV' heuristics and 'mac' inference the components should be laid out as:

bbbbbb. .... aa.eeee aa.eeee aa..... aa..gg. aa..gg.

using the coordinates: {'a': (0, 0), 'b': (0, 6), 'e': (3, 3), 'g': (4, 0)}

CircuitBoardCSP solved by backtracking in runtime: 0.000356 seconds using the following: degree_heuristics, 'MRV', 'LCV' heuristics and 'forward checking' inference the components should be laid out as:

bbbbbb. .... eeee.aa eeee.aa .....aa ..gg.aa ..gg.aa using the coordinates: {'a': (5, 0), 'b': (0, 6), 'e': (0, 3), 'g': (2, 0)}

CircuitBoardCSP solved by backtracking in runtime: 0.000423 seconds using the following: degree_heuristics, 'MRV', 'LCV' heuristics and 'mac' inference the components should be laid out as:

bbbbbb. .... eeee.aa eeee.aa .....aa ..gg.aa ..gg.aa

using the coordinates: {'a': (5, 0), 'b': (0, 6), 'e': (0, 3), 'g': (2, 0)}

c) Let (x, y) be an arbitrary location of a variable X (ie the lower left corner of a component) on the board. Then the domain of X corresponding to a component of width w and height h, on a circuit board of width n and height m is given by all (x, y) locations on the board such that x+w <= n and y+h <= m.

The code converts an arbitrary location (x, y) on a board of size n by m to an integer location by the formula: y*n + x. The formula works in the given manner: given that the width of the board is n, there are n spots (indices) on a given row. Since the location (x, y) has y+1 rows, there must be at least y*n spots (indices) to get to this location. On the row that this location is on, x indications the column number where the location is, implying that we have to add x many spots to y*n to get the number of spots of how many rows up on the board has been covered.