**CMSI 282 – Classwork 5**

**Instructions:**

This worksheet gives you some important practice with the fundamentals of CSPS!

* Provide answers to each of the following questions and write your responses in the blanks. If you are expected to show your work in arriving at a particular solution, space will be provided for you.
* Place the names of your group members below:

**Group Members:**

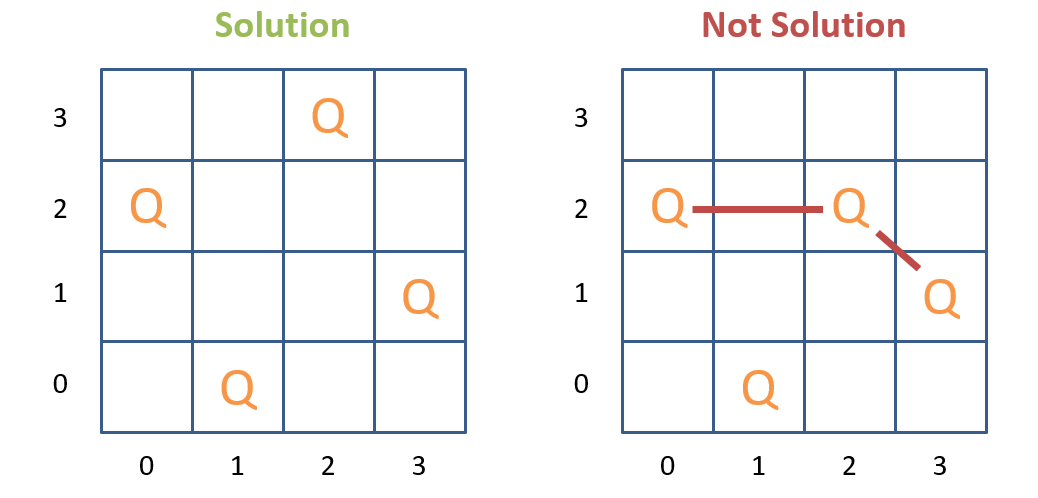
1. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
2. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
3. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
4. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
5. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem 1 – CSP Formalization & Backtracking**

Time for another rite of passage in this Tour de Algorithms we call 282: The N-Queens Problem.

*The N-Queens Problem* is a CSP in which the goal is to place chess queens on an chess board such that no queen can capture any other. For those unfamiliar with chess, Queens are the most powerful piece that, on any given turn, can move as many tiles as desired in horizontal, vertical, and diagonal directions. A Queen is capturable if an attacker can move from its tile to that of in a single turn.

Below, observe (left) a solution to the 4-Queens problem and (right) constraints violated on that same problem (the Queen at can capture both the one at and ).



In these problems, we have Queen variables to place with the implicit constraint that none may capture one another (too verbose to list explicitly).

* 1. In terms of representing the domains of each of the variables, why might it be wasteful to represent them as sets of tuples for individual board positions? Can you suggest a better representation for these variables’ values?

|  |
| --- |
|  |

* 1. Will constraint propagation *preprocessing* be of any use to for reducing variable domains in this problem? What about constraint propagation *during* backtracking?

|  |
| --- |
|  |

OK spoilers to 1.1 – you didn’t read ahead, did you? Be honest! – Suppose we represent the partially specified assignments during backtracking on the N-Queens problem as a tuple of queens in which the tuple index specifies each Queen’s column, and int contents specify each Queen’s row.

For example: denotes a partial specification in which (because it’s at the 0th index and the value there is a 2), , and have yet to be assigned.

* 1. Complete the backtracking recursion tree *with forward checking during search* that would solve the 4-Queens problem using the above partial-specification-notation. [Hint: the tree will only have 9 states before a solution is found].

|  |
| --- |
|  |

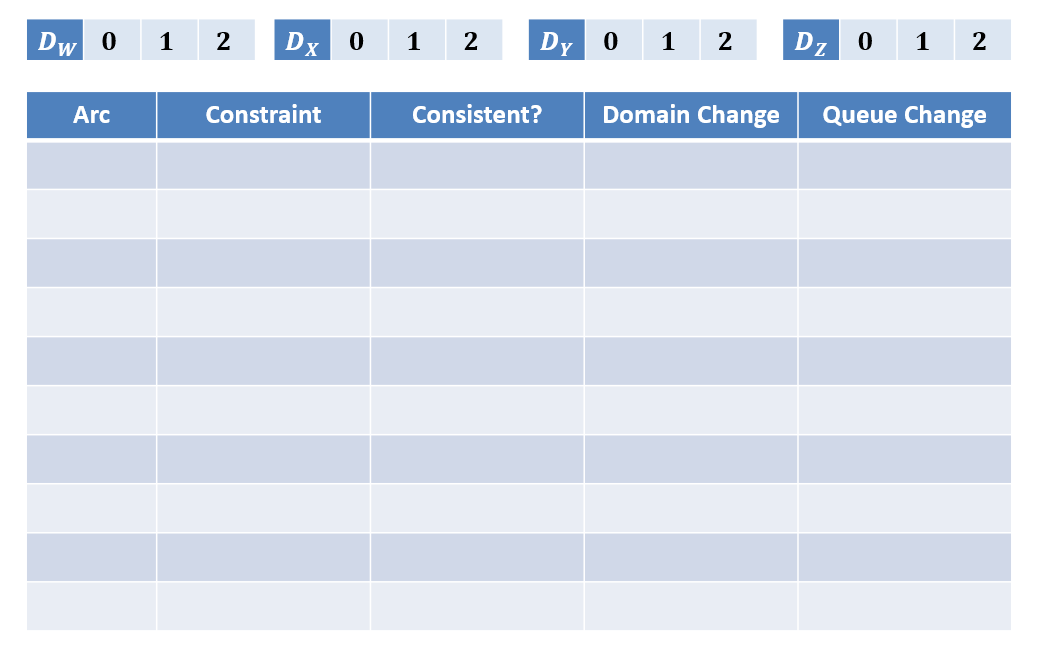
**Problem 2 – Constraint Propagation**

Since it’s relevant to your upcoming assignment, let’s talk about a nice numerical example; we’ll do something similar with Dates in HW 5, but for now, let’s just stick with numbers.

* *Variables:*
* *Domains:*
* *Constraints:*

**2.1.** Draw the *constraint graph* associated with the CSP above.

|  |
| --- |
|  |

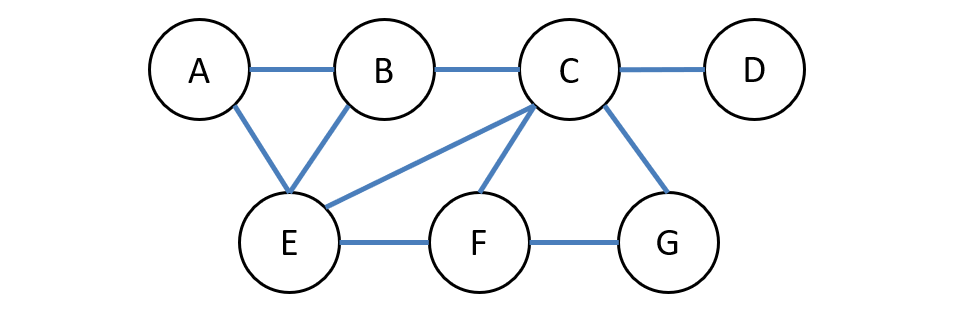
**2.2.** Perform the AC-3 algorithm for constraint propagation to restrict the domains of the above. The Queue of arcs and Domains have been *initialized* for you. Fill in the table, the queue, and cross off values in each domain as you perform the steps of AC-3.

|  |
| --- |
| **Arc Queue (Front -> Back):** |

**Problem 3 – Tree-Structured CSPs & Cutset Conditioning**

Let’s get back to basics with some good old Map Coloring. As a refresher, in the Map Coloring problem, the goal is to assign one of colors to variables in some geographically connected setting such that no two adjacent variables share the same color.

Suppose we have the following constraint graph associated with a Map Coloring problem in which edges denote adjacency between variables, and we have colors to work with:



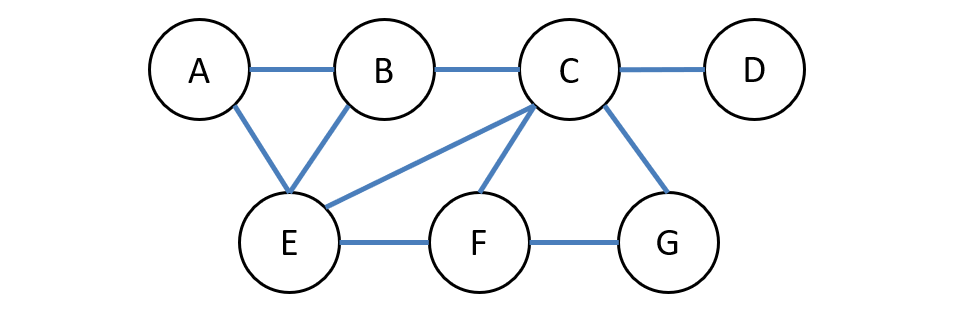
**3.1.** Suppose, during backtracking, we begin by assigning and are performing *forward checking*. Which variable would be wise to assign to next, and why?

|  |
| --- |
|  |

**3.2.** Suppose, during backtracking with forward checking, we begin by assigning and have decided to assign to next. At this point, the domain of will have been reduced to from forward checking; which of these values would we be wise to try assigning to next, and why?

|  |
| --- |
|  |

Repeating the Constraint Graph from the previous page for convenience:



**3.3.** Find a *minimal* cutset such that the remaining variables are tree-structured.

|  |
| --- |
|  |

**3.4.** *Condition* on your cutset and adjust the remaining tree’s domains accordingly:

|  |
| --- |
|  |

**3.5.** Perform the *Directed Arc Consistency Tree CSP Solver* like done in class in the table here:

