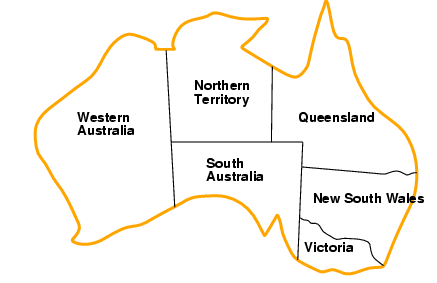
***# Questions = 5. Total # points = 75.***

**1. [15 points]** In the 5-queen problem, the goal is to place 5 queens on a 5 x 5 board such that no two queens attack each other horizontally, vertically or diagonally. Formulate the 5-queen problem as a binary Constraint Satisfaction Problem (CSP). Let each column has exactly one queen and variables Q1, Q2, Q3, Q4 and Q5 represent the position of the five queens in columns 1 to 5, respectively. (a) Specify the domain of each variable. (b) Specify the constraints among the variables. (c) Draw the constraint graph for this problem.

**2. [15 points]** Given the map-coloring problem as shown below, the goal is to color the map such that no two adjacent regions have the same color. There are three colors available: red (R), green (G), and blue (B). People in the region Victoria (V) prefer the red color and we will assign red to variable V first. Use the Backtracking-Search algorithm for CSP in Figure 1 below to color the remaining regions of the map. Note that the algorithm below does not contain the INFERENCE function that is in the algorithm in the lecture slides and book. In the backtracking search, use the minimum remaining value (MRV) and the degree heuristics to select the next variable, and use the least constraining value heuristic to order domain values. Draw the search tree produced and put down the order nodes are generated in the upper-right corner of a node. Write down the final solution (or final assignment.)



**Map of Australia**

**3. [10 points]** What are the time complexity and space complexity if the following algorithms are used to solve a constraint satisfaction problem (CSP)? Define the parameters you use. (a) The regular uninformed TREE SEARCH algorithm with depth-first search strategy. (b) The Backtracking Algorithm in Figure 1 below.

**4. [20 points]** We would like to formulate the following cryptarithmetic problem as a constraint satisfaction problem (*CSP*), where each letter stands for a *different* digit. The aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct, with the added restrictions that A ≠ 0 and C ≠ 0. Auxiliary variables  and are introduced for representing carry-overs.

**CD**

**+ CD**

**-------------**

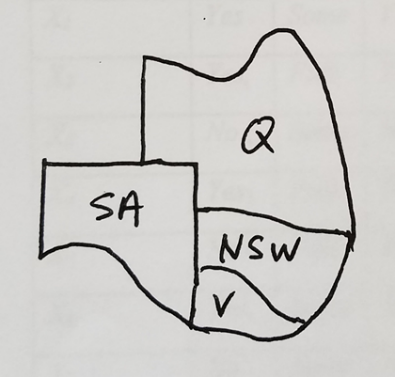
**ABC**

1. Specify the constraints for the *CSP* as a set of equations relating variables A, B, C, D and auxiliary variables  and .
2. Specify the domain values for variables A, B, C, D, and auxiliary variables  and .
3. Draw a constraint hypergraph to represent the constraint relationship between the variables and the auxiliary variables.
4. Suppose we have just assigned a value of 6 to the variable C and the current domain values of the other variables are as shown in the table below, apply *forward checking* one time and list the domain values of each variable after forward checking.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Variables** | A | B | C | D |  |  |
| **Initial** | 1 | 2,3,6,7 | 6 | 3,4,8,9 | 0,1 | 1 |
| **After FC** |  |  |  |  |  |  |

**5. [15 points]** Given the partial map of Australia as shown in the figure below, the goal is to color the map such that no two adjacent regions have the same color. There are three colors available: red (R), green (G) and blue (B). The table below shows the current domain values of the variables. Apply the arc consistency algorithm (AC-3) to reduce the domain of the variables. Assume that initially the *queue* contains only the arc. (a) Show the domain values of each variable after the AC-3 algorithm has stopped. (b) List all the arcs that are added and subsequently removed from the *queue* by the algorithm. List them in the order they are added to the queue. (**Note**: the line of code “**for** each in **do**” in the AC-3 algorithm below is different from the algorithm in the text book. Use the algorithm below for this problem.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | SA | Q | NSW | V |
| Current Domain Values | RGB | G | RG | RGB |
| After AC-3 |  |  |  |  |

****

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**function** BACKTRACKING-SEARCH(*csp*) **returns** a solution or failure

**return** BACKTRACK({ }, *CSP* )

**function** BACKTRACK(*assignment, csp*) **returns** a solution or failure

**if** *assignment* is complete **then return** *assignment*

*var* SELECT-UNASSIGNED-VARIABLE(*csp*)

**for each** *value* **in** ORDER-DOMAIN-VALUES(*var,assignment,csp*) **do**

if*value* is consistent with *assignment* **then**

add {*var* = *value*} to *assignment*

result BACKTRACK(*assignment, csp*)

**if** *result != failure* **then**

**return** *result*

remove {*var=value*} from *assignment*

**return** *failure*

**Figure 1. Backtracking Algorithm for CSP**