CS 4613 **Fall 2021**
Project 2: Map Coloring **E. K. Wong**

Total number of points = 100. **The implementation of the *INFERENCE* function with *Forward Checking* as described below is optional.** If you implement it, you will get 20 points of extra credits on top of the 100 points for the project.

Project Description: Implement the Backtracking Algorithm for CSPs in Figure 5 below to color a map so that no two adjacent regions have the same color. The regions in the map, the colors to be used and the region adjacency information are specified in an input file.

Implement the function *SELECT-UNASSIGNED-VARIABLE* in the algorithm by first using the *minimum remaining values* heuristic, then followed by the *degree* heuristic. If there are more than one variables left after applying the *degree* heuristic, the algorithm can arbitrarily choose the next variable to work on. You do not need to implement the *least constraining value* heuristic in the *ORDER-DOMAIN-VALUES* function; simply order the domain values in the same order given in the input file. For the *INFERENCE* function, implement *Forward Checking* as covered in the lectures.

Your program will read in the specifications for the CSP from an input text file and produce an output text file that contains the solution. Figure 1 below contains the format of the input file. The first line contains the number of variables (or regions) N and the number of domain values (or colors) d for the variables. The second line contains the variable names, with each variable consists of one to three letters. The third line contains the list of domain values (or colors), with each value consists of a single letter. This is followed by an $N \times N$ array. Element (i,j) in the array has a value of 1 if regions i and j are adjacent to each other; it has a value of 0 otherwise. Index i is for labelling the rows and index j is for labelling the columns. As an example, Figure 2 contains the content of the input file for the Australia map coloring problem (Input file 1.) As shown in Figure 3, labelling of the rows with variable names goes from top to bottom and labelling of the columns with variable names goes from left to right. The labelling follows the same order the variables appear in the input file. In your program, you will need to map the variable names to integers when accessing the array. Figure 4 contains the format of the output file. The output contains N lines, specifying the values of the N variables in the final assignment.

Team: You can work on the project alone or in a team of two persons. You can discuss with your classmates on how to do the project but every team is expected to write their own code and submit their own program and report.

Testing your program: Two input test files will be provided on BrightSpace for you to test your program.

Recommended languages: Python, C++/C and Java. If you would like to use a different language, send me an email first.

Submit on BrightSpace by the due date:

1. A text file that contains the source code. Put comments in your source code to make it easier for someone else to read your program. Points will be taken off if you do not have comments in your source code.
2. The output text files generated by your program. Name your output files *Output1.txt* and *Output2.txt*.
3. A PDF file that contains instructions on how to run your program. If your program requires compilation, instructions on how to compile your program should also be provided. Also, copy

and paste the output text files and your source code onto the PDF file (to make it easier for us to grade your project.) This is in addition to the source code and output files that you are required to submit separately (as described in 1 and 2 above.)

Only one partner in a team needs to submit, but write both partners' names in the source code and PDF report.

$N\ d$	//Number of variables (regions); number of domain values (colors.)
$x_1\ x_2\ \ldots . x_N$	//List of variable names; each variable name consists of 1 to 3 letters.
$v_1\ v_2\ \ldots v_d$	//List of domain values; each value is a single letter.
n n n ….n	//An $N \times N$ array for specifying the constraints; element (*i*,*j*) has a
n n n ….n	// value of 1 if regions *i* and *j* are adjacent to each other; it
……..….	// has a value of 0 otherwise.
n n n … n	

Figure 1. Input file format.

```
6 3
NSW NT Q SA WA V
R G B
0 0 1 1 0 1
0 0 1 1 1 0
1 1 0 1 0 0
1 1 1 0 1 1
0 1 0 1 0 0
1 0 0 1 0 0
```

Figure 2. Input file for the Australia map coloring problem (Input file 1.)

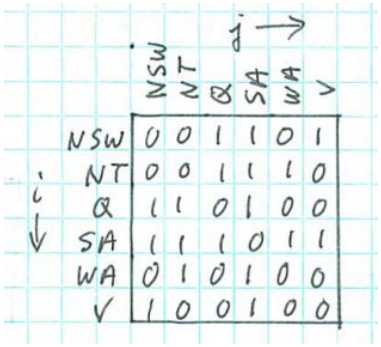

Figure 3. Labelling of the rows and columns of the array.

$x_1 = v_{x_1}$ //Variable name followed by "=" and its assigned value.

$x_2 = v_{x_2}$

$x_N = v_{x_N}$

Figure 4. Output file format.

```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
  return BACKTRACK(csp, { })

function BACKTRACK(csp, assignment) returns a solution or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp, assignment)
  for each value in ORDER-DOMAIN-VALUES(csp, var, assignment) do
    if value is consistent with assignment then
      add {var = value} to assignment
      inferences ← INFERENCE(csp, var, assignment)
      if inferences ≠ failure then
        add inferences to csp
        result ← BACKTRACK(csp, assignment)
        if result ≠ failure then return result
        remove inferences from csp
      remove {var = value} from assignment
  return failure
```

Figure 5. The Backtracking Algorithm for CSPs.