COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS
DEPARTMENT OF APPLIED INFORMATICS

Introduction to Artificial Intelligence

Exercise 5 - Constraint-Satisfaction Problem (CSP)

October 19, 2022

CONSTRAINT-SATISFACTION PROBLEM

The task is to implement a classic example of CSP (Constraint-Satisfaction Problem) - coloring the map of Australia (and other countries) using a chosen heuristics.

Program:

The most important class is **MapCSP** in the file *map_csp.py*, which contains:

- possible colors (list *self.color_options*),
- list of states (list *self.states*),
- coloring of individual countries (dict *self.colors*: country -> colour), after initialization the colors are set to *None*,
- functions *has_color*, *get_color*, *set_color*, *del_color* in order to color the map,
- helper functions *print_map*, *same_colors(state1, state2)*, *all_colored*.

There are several places in the code, where you need to write your code. Each of them is marked as usually.

- **can_set_color(state, color)** - returns *True* if we can color the state *state* with the color *color* without breaking the rules - none of the neighbours can have the same color as the *state*. The function does not change the colour!
- **select_next_state()** - chooses the next state to be colored. This is the place for your chosen heuristics.
- **color_map()** - this function colors all the states in a way, that none of two neighbours have the same color. The function returns *True* if it finds a valid coloring, otherwise returns *False* (when there is no valid coloring).

In **maps.py** are defined classes *AustraliaMap*, *USSRMap*, *USAMap* and *WorldMap*, which contain individual maps of sizes (7, 18, 51 and 248 states), and a wrapper class *ImpossibleMap*, which changes a given map so that there is no valid coloring.
Task 1 (0.5p): Complete the aforementioned functions to solve the basic CSP coloring, using backtracking (without inferences).
Task 2 (0.5p): Use any of the heuristics defined on the lecture for picking variables (*Minimum Remaining Value* / *Most Constraining Variable* = *Degree Heuristics* / ...) or a heuristic to choose a value (*Least Constraining Value* / ...). For our data *MRV* works the best, however all of the above

should improve the coloring speed. In the code clearly **mark and comment the heuristic you chose**.

Approximate run-times, when you have a correct algorithm and a good implementation:

- Australia, USSR: < 10ms, with or without heuristics.
- USA: < 3s without heuristics, < 50ms with any heuristics.
- World: don't even try this without heuristics (probably won't finish), with MRV you can get under < 100ms.

Submit only *map_csp.py*.