Exercises - Home project 3

Problem [CSP on planar graphs]

We will implement the classic CSP (Constraint-Satisfaction Problem) of coloring planar graphs corresponding to maps.

Program Description:

The key class in the program is `MapCSP`, located in the jupyter notebook `Exercises - Home project 3.ipynb`. This class contains:

- **Possible colors**: A list of available colors (`self.color_options`).
- **List of states**: A list of states to be colored (`self.states`).
- **State borders**: A dictionary mapping each state to its neighbors (`self.neighbours` : state -> list of neighbors).
- **State colors**: A dictionary assigning a color to each state (`self.colors` : state -> color). Initially, all states have the color `None`.

Key methods of the class include:

- `has_color`, `get_color`, `set_color`, and `del_color` for managing the colors of states.
- Auxiliary functions:
  - `print_map` : To print the map.
  - `same_colors(state1, state2)` : Checks if two states have the same color.
  - `all_colored` : Checks if all states are colored.

Tasks:

There are several parts of the program you need to implement. These are marked clearly in the code:

1. `can_set_color(state, color)` :
   - Returns `True` if the state `state` can be colored with `color` without violating the constraints (no neighboring state with the same color).
   - This function does not modify the map's coloring.
2. `select_next_state()` :
   - Chooses the next state to be colored.
   - This function is where you can apply heuristics for variable ordering.
3. `color_map()` :
   - Colors the states such that no two adjacent states have the same color.
   - Returns `True` if a valid coloring is found, or `False` if no valid coloring exists.

Additional Files:

- there are classes (do not change anythin there) representing different maps:
  - `AustraliaMap`, `USSRMap`, `USAMap`, and `WorldMap`, with increasing map sizes (7, 18, 51, and 248 states, respectively).
  - `ImpossibleMap` : A wrapper class that modifies a map to make it impossible to color.

Assignment Goals:

1. **Task 1 (this will allow you to solve the first three maps)**:
   - Implement the mentioned functions to solve the map coloring CSP using **basic backtracking** (no inference).
2. **Bonus Task 2 (needed for the last map)**:
   - Use a heuristic for variable ordering (e.g., Minimum Remaining Value (MRV), Most Constraining Variable = Degree Heuristics, etc.) or value selection (e.g., Least Constraining Value). You have to look up on the internet what those heuristics do!
   - For this specific problem, MRV is the most efficient heuristic, but others work well too.
   - Clearly indicate which heuristic you used in the code and comment on any helper functions you add.

Performance Benchmarks:

If your algorithm and implementation are correct and efficient:

- **Australia, USSR (3pts each)**: < 10ms, with or without heuristics.
- **USA (4pts)**:
  - < 3 seconds without heuristics.
  - < 50ms with any heuristic.
- **World (5pts)**:
  - Do not attempt without heuristics.
  - With the MRV heuristic, the runtime should be < 100ms (depending on implementation efficiency).

Total points (10 + 5 Bonus)