Artificial Intelligence Assignment -II
Map Coloring

- ***Abhinav Rai (B14CS002)***
- ***Mahak Jain (B14CS042)***

Problem Statement

Generate a random map and a fixed number of colors, assign a color to each node in the map in such a way that no two adjacent regions have the same color.
Map Coloring can be visualized as constraint graph with -

- Nodes = variables
- Edges = constraints

Input Parameters

1. Number of nodes in the map
2. Maximum number of colors to be used to color the map
3. Average Density factor of the map
4. Maximum steps (Only in case of Local Search)

Heuristics Used

Variable Ordering - Most Constrained Variable (MRV) & Degree Heuristic

- We used **MRV** heuristic to pick the variable to be assigned. MRV picks a variable that is most likely to cause a failure soon, thereby pruning the search tree. If some variable X has no legal values left, the MRV heuristic will select X and failure will be detected immediately—avoiding pointless searches through other variables.
- MRV heuristic doesn't help at all in choosing the first node to color. Thus, we used **Degree** heuristic as tie-breaker. It attempts to reduce the branching factor on future choices by selecting the variable that is involved in the largest number of constraints on other unassigned variables.

Value Ordering - Least Constrained Value(LCV)

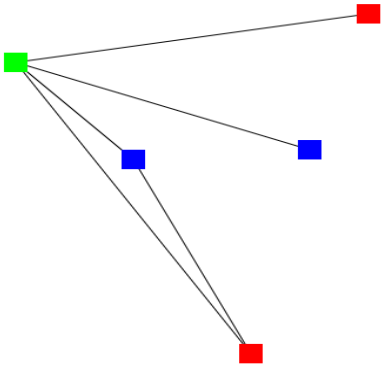
- **LCV** prefers the value that rules out the fewest choices for the neighbouring variables in the constraint graph. This heuristic is trying to leave the maximum flexibility for subsequent variable assignments.

Different Approaches

1. **Simple Backtrack -** The BACKTRACK search is a depth-first search that chooses values for assignment of one variable at a time and backtracks when a variable has no legal values left to assign.

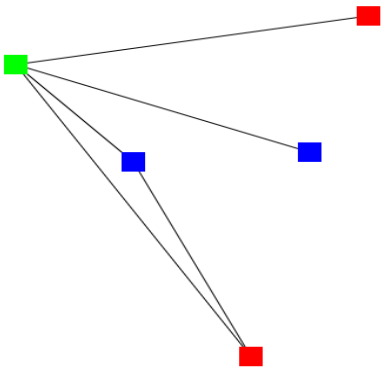
- It repeatedly chooses an unassigned variable using MRV heuristic and in case of tie, use DEGREE heuristic to choose the variable
- Then it picks the color which is least constrained using LCV heuristic and assign it to the chosen variable.
- If any inconsistency is detected, then BACKTRACK returns failure, causing the previous call to try another value.

It takes a lot of time in coloring as it tries all possible assignments to every variable till a solution is reached.



2. **Backtrack with Forward Check -** In this approach, in addition to BACKTRACK search, before assigning a variable we perform a FORWARD-CHECKING.
    - It establishes ARC-CONSISTENCY of the variable node with all its neighbouring nodes. For this, we remove the assigned color from the domains of neighbouring nodes which are inconsistent with the assigned color of the variable.
    - If this leads to a state which is not solvable, then a different value is tried for that variable node.

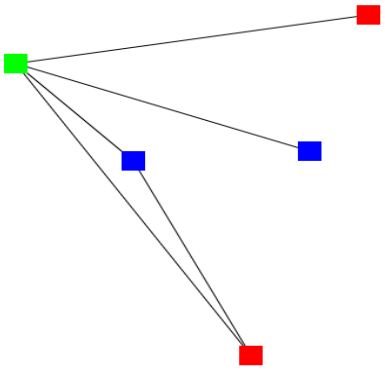
    It takes time lesser than Simple Backtrack in coloring as it prunes some of the assignments which Simple BACKTRACK approach considers.



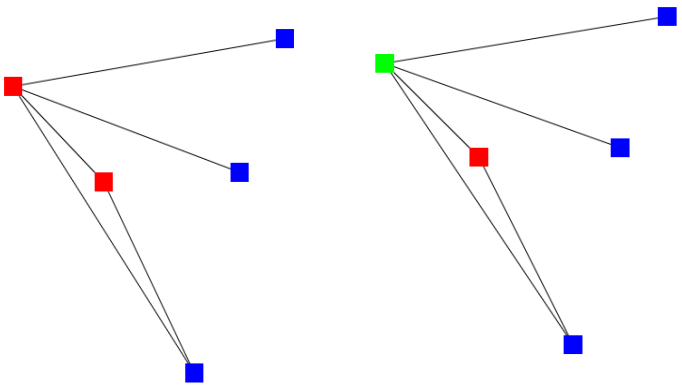
3. **Backtrack with MAC (Maintaining Arc Consistency) -** Although forward checking detects many inconsistencies, it does not detect all of them.

- This approach makes all the other variables also arc-consistent along with the neighbouring variables.

This approach usually takes time more than forward checking in coloring as it is a much stronger inference as compared to FORWARD-CHECKING due to the overhead of making all nodes ARC-CONSISTENT.



4. **Local Search using MIN_CONFLICTS heuristic -** In this approach, we follow the below steps -
    - First we color the complete map randomly. Typically, the initial guess violates several constraints.
    - The point of local search is to eliminate the violated constraints. This approach takes maximum number of steps as input parameter.
    - In each step, we randomly choose a variable node that is inconsistent and assign a new color to it.
    - In choosing a new color for a variable, the heuristic used is MIN-CONFLICTS i.e. to select the color that results in the minimum number of conflicts with other variables.It works great for some problems but may fail in some problems.
    - The algorithm ends if a consistent solution is not reached in specified steps.



Random Coloring Modified Solution

Observations

- The following table summarizes the time taken by each approach used for Map coloring with different number of nodes.

Approaches	***10 Nodes***	***50 Nodes***	***100 Nodes***
Simple Backtrack	3.2 ms	4.327 sec	About an hour!
Forward Checking	3.0 ms	6 ms	1 min 20 sec
MAC	3.4 ms	9 ms	1 min 50 sec
Local Search	3.2 ms	2.3 ms	7.3 ms

- Forward Check was found to be the most efficient complete algorithm.

- The local search was found to be very efficient for Map coloring problem (But not complete). It took around 7 steps to color a map of 10 nodes and around 30 steps to color a map of 100 nodes.

- Comparison on basis of time taken ***Forward Checking < MAC < Backtrack***
  Simple backtrack algorithm takes much more time than other 2 because of the absence of inferences. Forward Checking takes less amount of time because in this we are just reducing the domain of the neighbouring nodes while in MAC, the domain of all the nodes in the graph can be reduced which will obviously take huge amount of time.

- The above diagrams show that first three approaches color the map in similar way, only the difference is the time the approaches to take to select the variable nodes and colors. As Local search is Random Algorithm, coloring is different each time.