HOMEWORK 02

Courses `CSC14003`: Intro to Artificial Intelligence
18CLC6, FIT - HCMUS.
`09/08/2020`

This is a team homework for 2 students, but for some reason - can not form a team, so I do it as a individual assignment :

- `18127231`: Đoàn Đình Toàn (GitHub: @t3bol90)

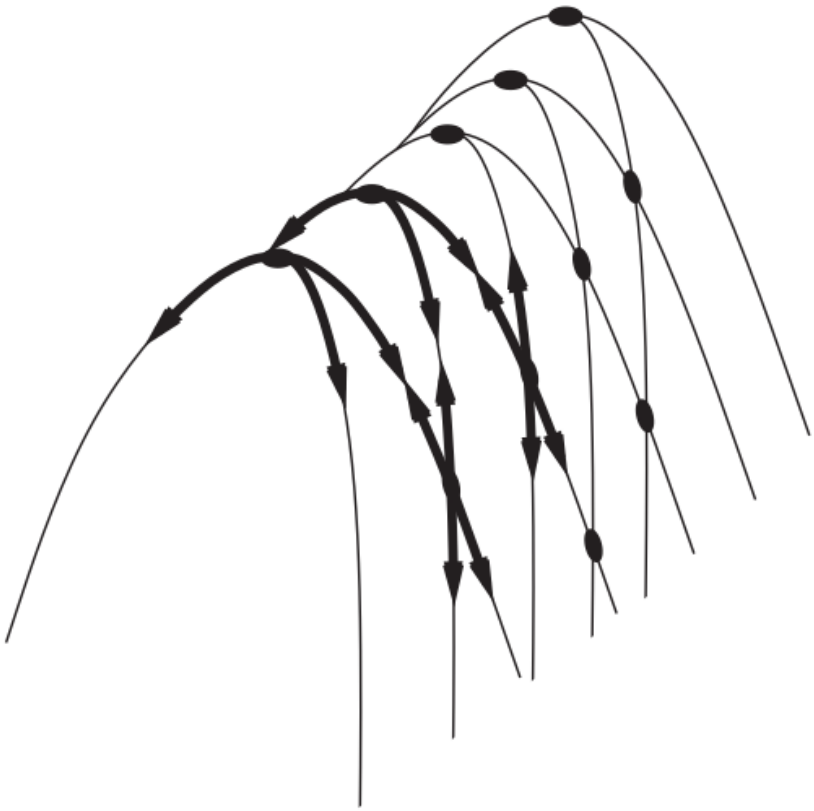
Problem 1. (3.0pts) [Local search]

Answer the following questions about local search algorithms

- a) Which places in the state space make regular hill climbing fail to the optimal solution?

  Answer: Regular hill climbing fail to find the optimal solution when searching space contain many local peaks (local maxima or minima if you are go in down hill). It lead the hill climbing search fall to stuck in local peaks many times. Another problem call ridges, a sequences of local maximas, making it difficult for navigating. And plateaux, the flat state-space, it often make hill climbing gets lost.

  Eg: the picture bellow show that search space contrain many peaks as ridges.



  This problem can be solve by using **Random-restart hill climbing**, **First-Choice Climbing** or **Stochastic Hill Climbing**...

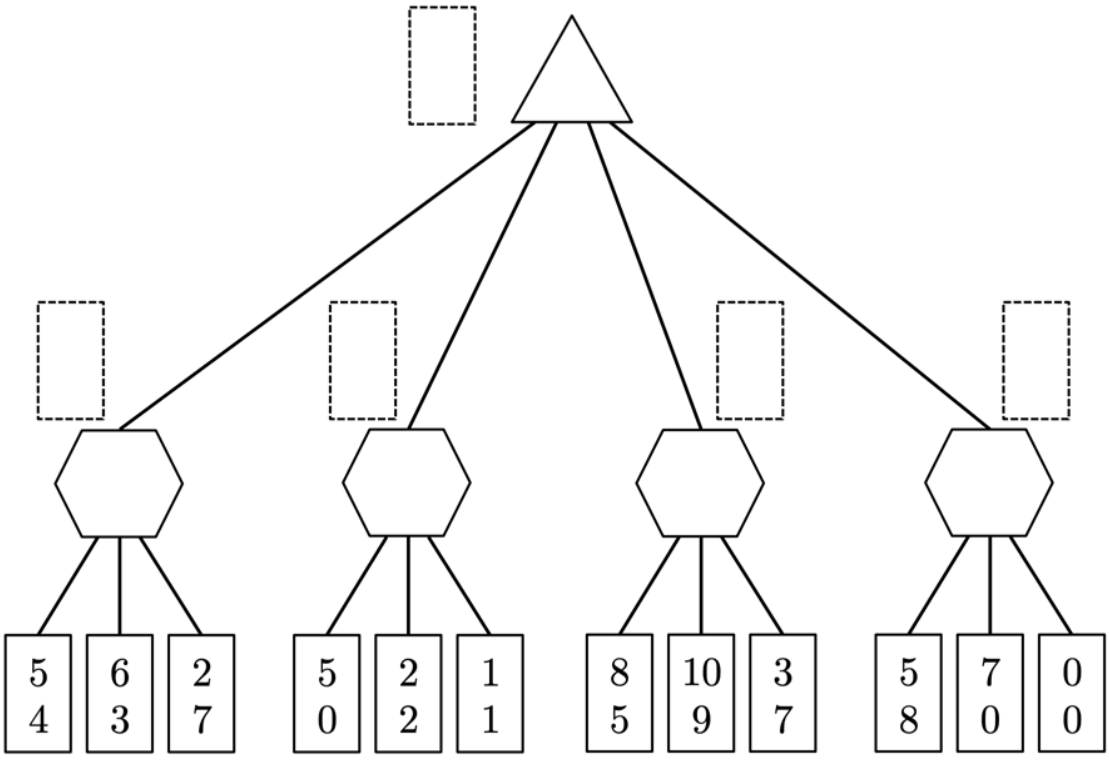
- b) Are variations of hill climbing (such as Random-restart hill-climbing, local beam search, etc.) guarantee complete and/or optimal? Briefly explain.

Answer: It depend on which variation you are using. Random-restart hill climbing is complete if infinite (or sufficiently many tries) are allowed. Local beam search is complete if k is infinite or large enough to cover all of case. In a different way, First-choice hill-climbing or Stochastic hill-climbing is not complete - it just less likely to get stuck. The optimal of such algorithms depend on the searching space. Convex space can be found optimal solutions by hill-climbing, but some complex space can not be optimal.

- c) It is possible to get stuck in a local maximum in simulated annealing. Is it TRUE or FALSE? Briefly explain.
  Answer: In theory, simulated annealing can advoids local maximum by escape from it :). Allowing 'worse' move, lesser quality to be taken some of the time. Simulated annealing will find a global optimum with probability approaching 1 if you lower T slowly enough. So the statement above is true, it is possible to get stuck in local maximum if you lower T too quickly! If you are using gradient descent, these problem come when step size is so large.

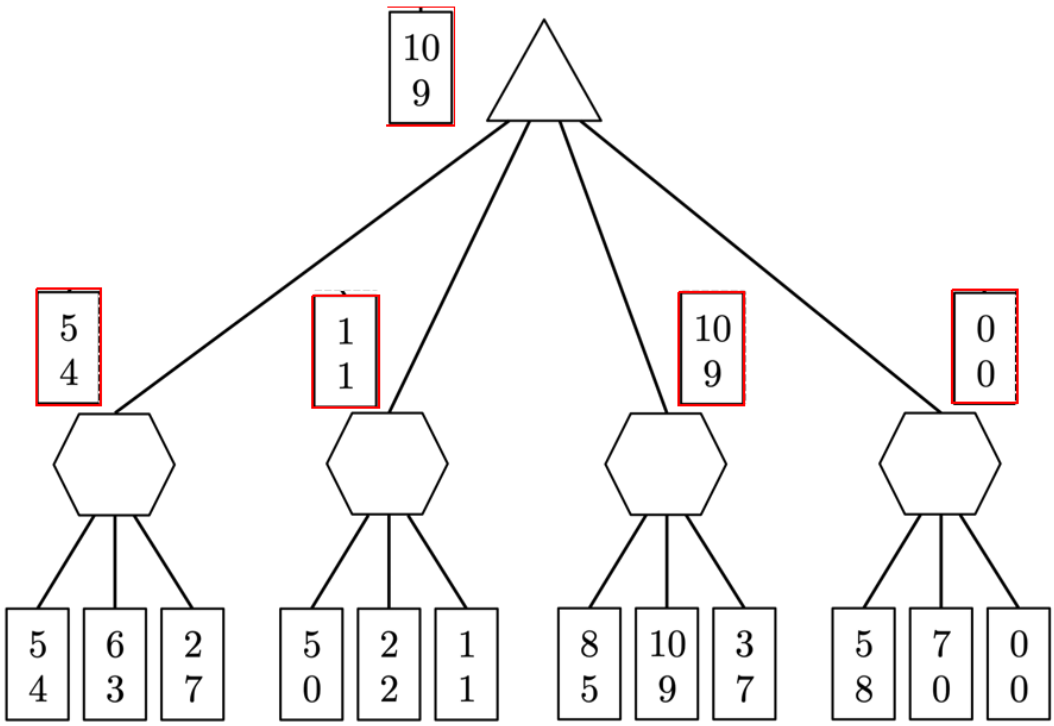
Problem 2. (3.0pts) [Adversarial Search]

You are an excellent programmer and you program a robot to play a scoring game with you. You and your robot both have your own score. You are trying to maximize your score and do not care about your robot' score. On the other hand, your robot tries to minimize the absolute difference between the two scores. In case of a tie, the robot prefers a lower score. For example, the robot prefers (4,3) to (5,3) since the abs(4-3) is less than abs(5-3); it prefers (2,3) to (4,3) since a tie occurs and (2, 3) is lower than (4,3); and it prefers (7, 7) to (9, 9) because a tie occurs and (7, 7) is lower than (9, 9).



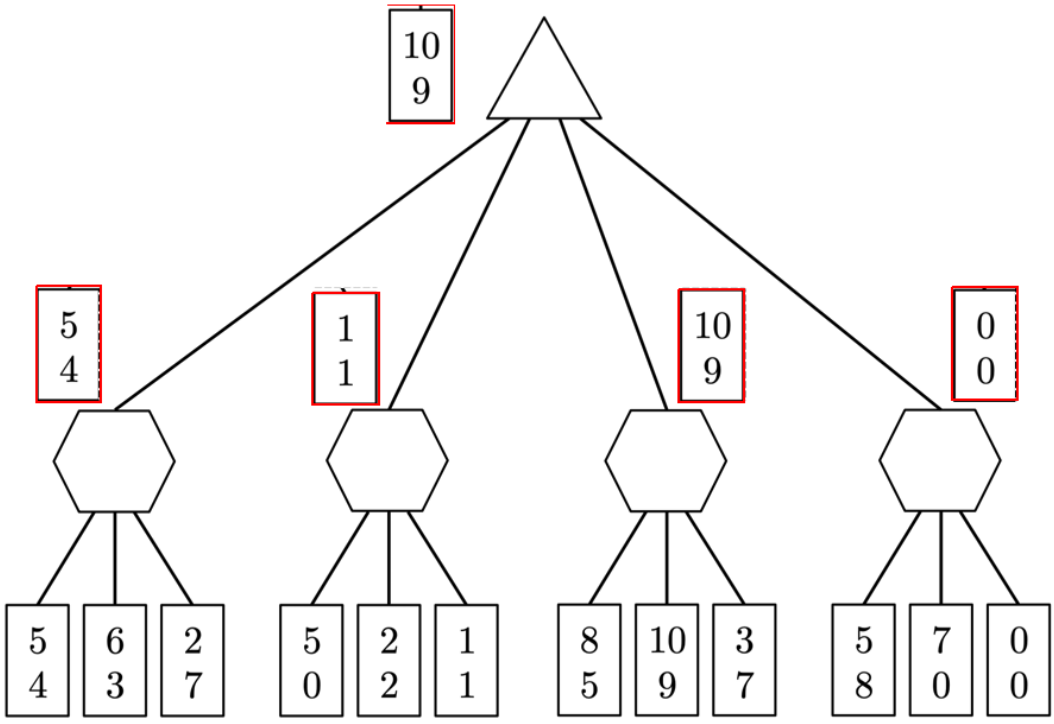
The above tree shows the game tree. The root node is your max node and the below nodes are the robot's nodes for four different actions. The leaf nodes are the scores with your score on top and the robot' scores are on the bottom.

- a) Fill in the dashed rectangles with the pair of scores preferred by each node of the game tree.



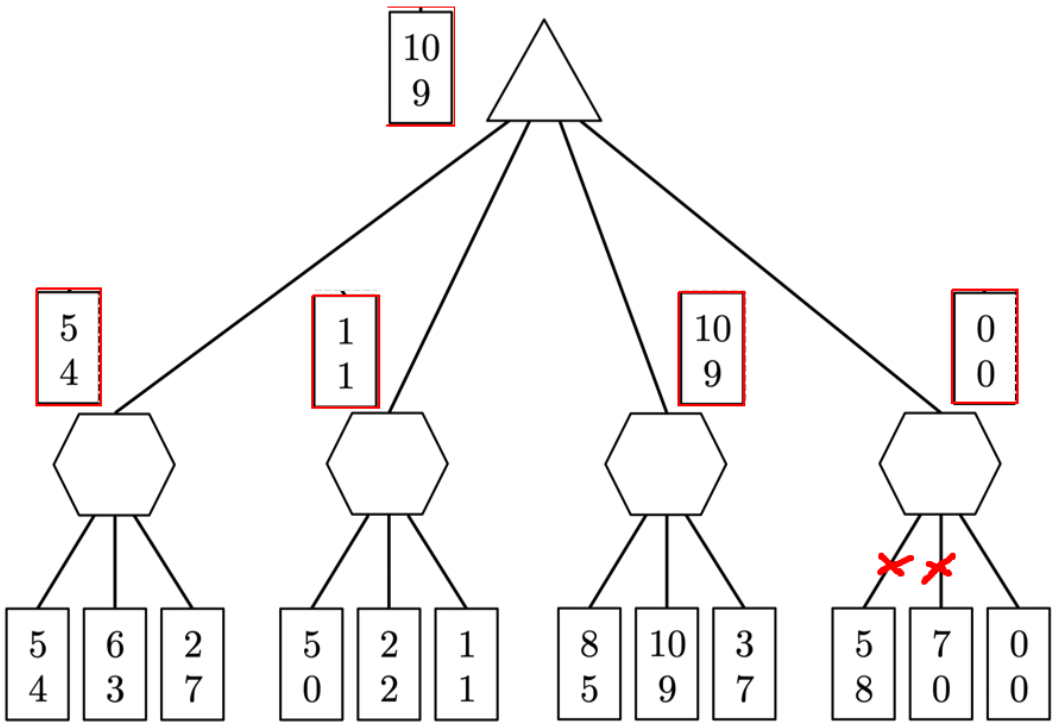
- b) You can save computation time by using pruning in your game tree search. Is it possible to prune any branch on the tree? If yes, please mark the branches with an 'X'.

  Answer: You can not prune any branch:

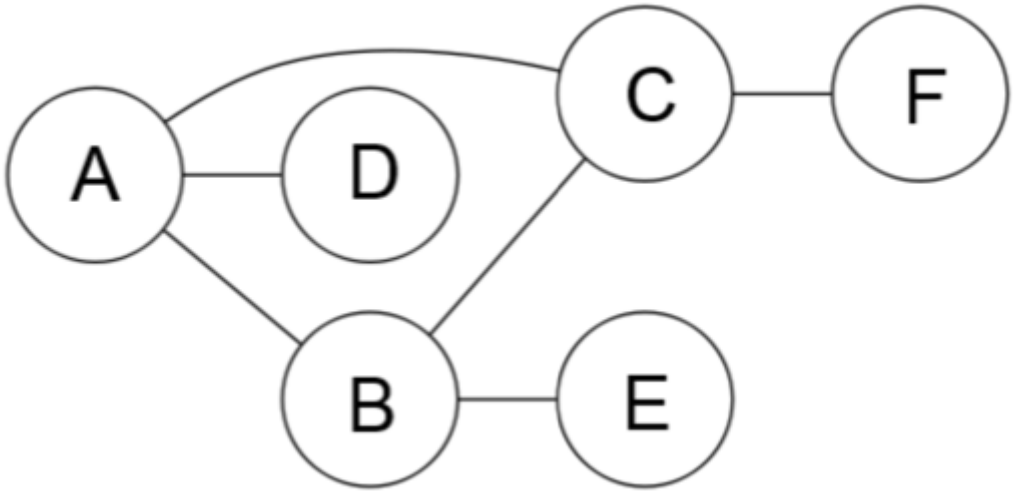


- c) You now have access to an oracle that tells you the order of branches to explore that maximizes pruning. On the copy of the game tree above, put an 'X' on line of branches that do not need be explored given this new information from the oracle.

  Explain: The min step (robot) always looking for [0,0] because [0,0] is the minima of min step. So if we can pick [0,0] at the beginning, 2 branchs can be pruned. The max step can not prune any branch because the maxima of max step is infinite.



Problem 3. (2.0pts) [Constraint Satisfaction Problem]



The graph above is a constraint graph for a CSP that only has binary constraints. Each node represents a variable. At the beginning, the variables have not been assigned any value.
Please answer the following questions, given that assigning values to the variables results in their domain being changed.

- a) A value is assigned to A. Which domains might be changed as a result of running forward checking for A?

  Answer: Forward checking affect on the arcs when A is a head. So B $\rightarrow$ A, C $\rightarrow$ A and D $\rightarrow$ A. Therefore, B, C and D might be changed.

- b) A value is assigned to A, and then forward checking is run for A. Then a value is assigned to B. Which domains might be changed as a result of running forward checking for B?

  Answer: Forward checking affect on the arcs when B is a head. So A $\rightarrow$ B, C $\rightarrow$ B and E $\rightarrow$ B. However, A is assigned, so jut C and E might be changed.

- c) A value is assigned to A. Which domains might be changed as a result of enforcing arc consistency after this assignment?

  Answer: Arc consistency can affect all of node/domains in graph if A is simply connected (have path) to another node. This graph is Connectivity Graph so it will be affect on all of domains. B, C, D, E and F might be changed as a result.

- d) A value is assigned to A, and then arc consistency is enforced. Then a value is assigned to B. Which domains might be changed as a result of enforcing arc consistency after the assignment to B?

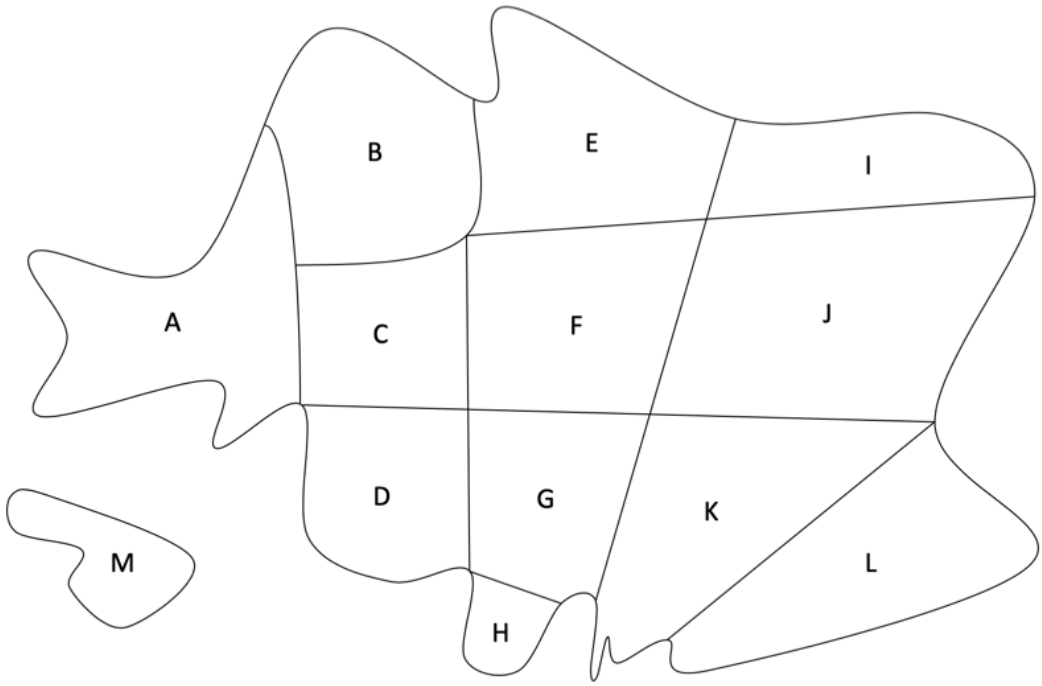
  Answer: After arch consistency is enforced on A, future assignments and arc consistency will not changed to A. So when enforcing arc consistency on B, it can not reach A and D. So C, E and F might be changed as a result.

Problem 4. (2.0pts) [Map coloring –Constraint Satisfaction Problem]

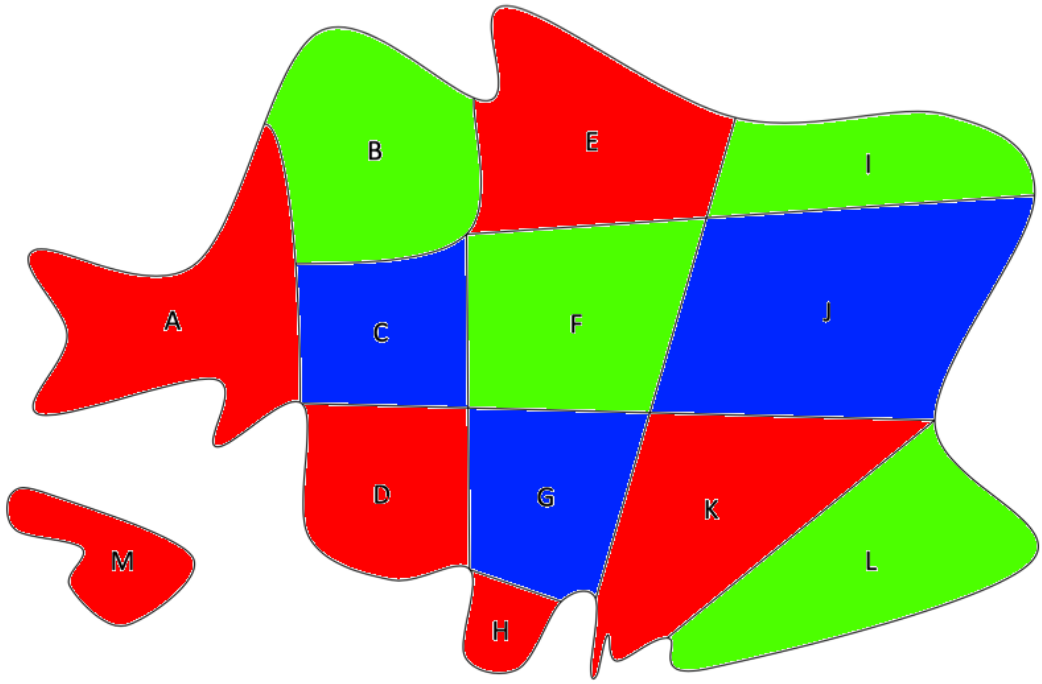
In the given map below, there are 14 regions corresponding to 14 capital letters (from 'A' to 'M').

- a) Please find the minimum number of colors needed to color the regions with the constraint that no two adjacent regions have the same color. You just need to state the minimum number and give one sample of the colors assigned to each of the regions that satisfy the constraint.

  Answer: Minimum number of colors needed to color the regions is 3 colors. Because It's a planar graph and A,B and C need at least 3 colors to label. So the answer is 3.

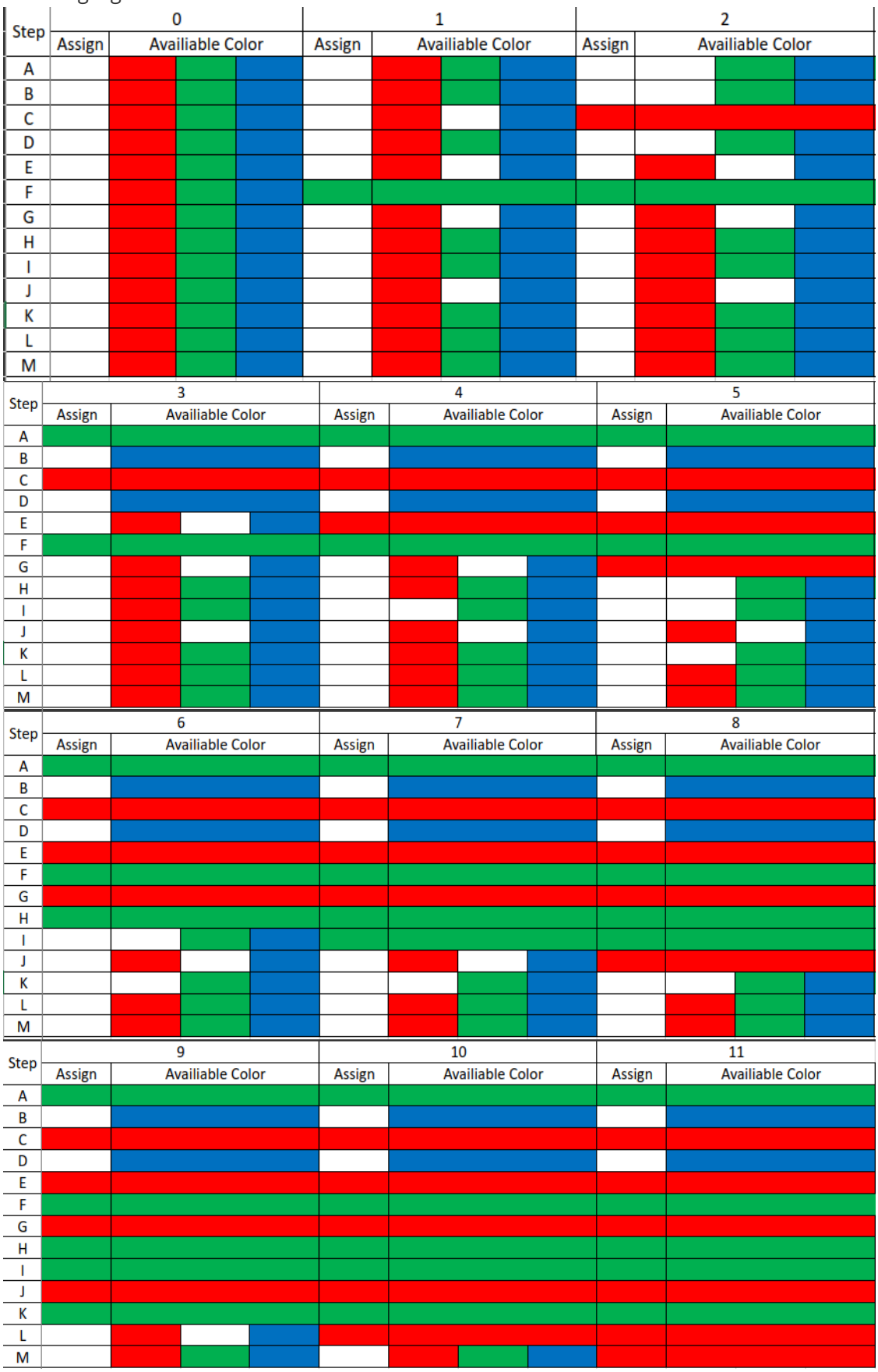


Graph can color with 3 colors as:

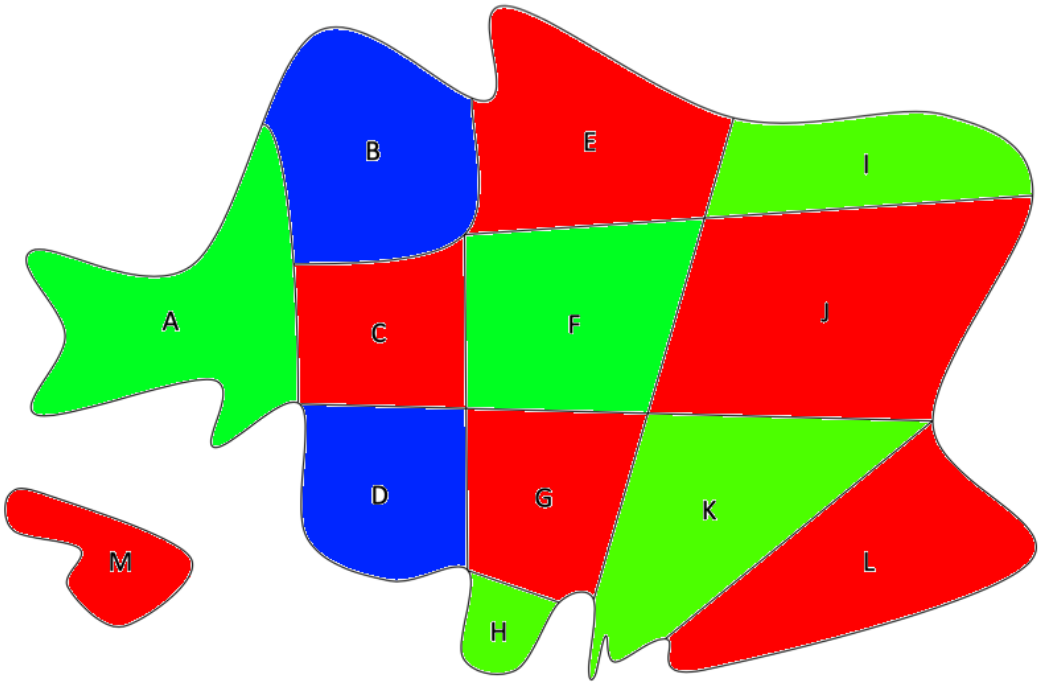


- b) Assuming that there are three colors: red, blue, and green. Initially, we can give every region one of the colors. It means that the color domains of each of the regions are {red, blue, green}. Then, we assign the region F to have green color. What is the result of the Forward Checking algorithm?

Answer: I using MRV (Minimum Remaining Values) as heauristic method for Forward Checking algorithm.



Final result:



====== END =====