Graph Coloring Problem using Backtracking:

To solve the problem follow the below idea:

The idea is to assign colors one by one to different vertices, starting from vertex 0. Before assigning a color, check for safety by considering already assigned colors to the adjacent vertices i.e check if the adjacent vertices have the same color or not. If there is any color assignment that does not violate the conditions, mark the color assignment as part of the solution. If no assignment of color is possible then backtrack and return false

Follow the given steps to solve the problem:

- Create a recursive function that takes the graph, current index, number of vertices, and output color array.
- If the current index is equal to the number of vertices. Print the color configuration in the output array.
- Assign a color to a vertex (1 to m).
- For every assigned color, check if the configuration is safe, (i.e. check if the adjacent vertices do not have the same color) recursively call the function with the next index and number of vertices
- If any recursive function returns true break the loop and return true
- If no recursive function returns true then return false

Time Complexity: $O(m^V)$. There is a total of $O(m^V)$ combinations of colors. The upper bound time complexity remains the same but the average time taken will be less.
Auxiliary Space: O(V). The recursive Stack of the graph coloring function will require O(V) space.

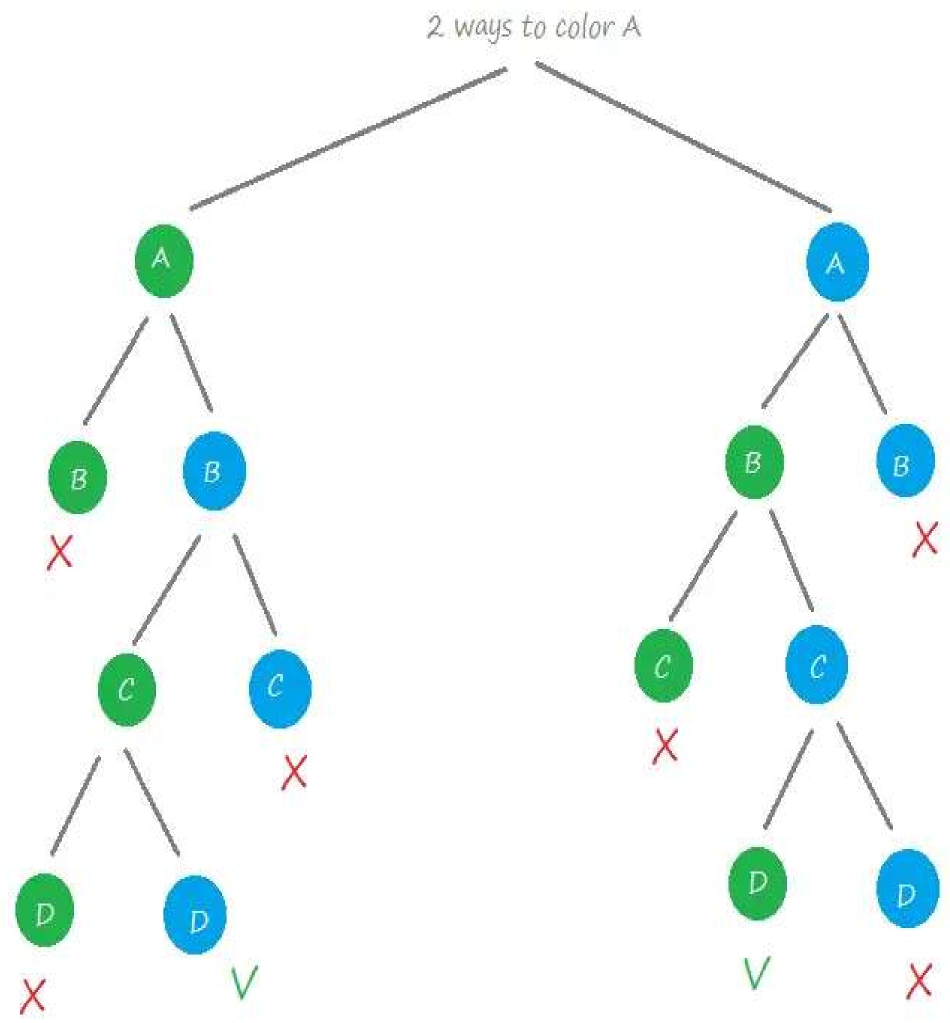
Example
The backtracking algorithm makes the process efficient by avoiding many bad decisions made in naïve approaches.

In this approach, we color a single vertex and then move to its adjacent (connected) vertex to color it with different color.

After coloring, we again move to another adjacent vertex that is uncolored and repeat the process until all vertices of the given graph are colored.

In case, we find a vertex that has all adjacent vertices colored and no color is left to make it color different, we backtrack and change the color of the last colored vertices and again proceed further.

If by backtracking, we come back to the same vertex from where we started and all colors were tried on it, then it means the given number of colors (i.e. ‘m’) is insufficient to color the given graph and we require more colors (i.e. a bigger chromatic number).



Graph coloring problem algorithm
Steps To color graph using the Backtracking Algorithm:

Different colors:
Confirm whether it is valid to color the current vertex with the current color (by checking whether any of its adjacent vertices are colored with the same color).
If yes then color it and otherwise try a different color.
Check if all vertices are colored or not.
If not then move to the next adjacent uncolored vertex.
If no other color is available then backtrack (i.e. un-color last colored vertex).
Here backtracking means to stop further recursive calls on adjacent vertices by returning false. In this algorithm Step-1.2 (Continue) and Step-2 (backtracking) is causing the program to try different color option.

Continue – try a different color for current vertex.
Backtrack – try a different color for last colored vertex.