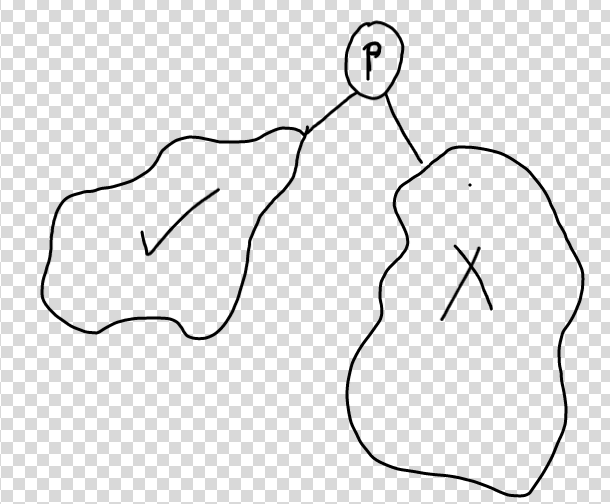
**Problem :** [**https://practice.geeksforgeeks.org/problems/m-coloring-problem-1587115620/1#**](https://practice.geeksforgeeks.org/problems/m-coloring-problem-1587115620/1#)

**Approach :**

-> DFS won’t work bcoz we might have to visit a visited node again, if the parent has to change the color.

-> E.g figure below : Suppose we visited all nodes in left subtree of ‘p’ and we were able to color all of them, but now assume we were not able to color all nodes in right subtree, so the process backtracks and tries a different color for ‘p’, hoping that now the right subtree nodes will be colored.

->But now all the left subtree nodes have to be traversed again, as when we visited them the last time, ‘p’ had a different color.But now bcoz its marked as visited it won’t be visited again.



-> If we follow procedure of marking a node as unvisited after its dfs completes,then program might fall into an infinite loop,so that too won’t help .Eg:



**Solution :**

Keep coloring nodes from 0 to n-1, but before coloring every node, check if it’s adjacent vertex is colored with the same color we are trying to assign to it. If so then try a different color.

If no color possible for some node ‘u’ , backtrack and try a different color for the parent ‘u-1’,if for parent also, not possible then set its color to -1, then backtrack to it’s parent ‘u-2’ to try different color and so on….

**Time Complexity : O(M^N) ,** M=no. of colors, N=no. of nodes

-> Each of the N nodes have ‘M’ choices for colours, so M\*M\*M,....N times = M^N

-> Off course some of the choices are invalid and would be discarded, but this is just the Upper bound on the Time.

**Code :** [**https://practice.geeksforgeeks.org/viewSol.php?subId=fe2a679f92c56a79f56603eda13561f7&pid=701374&user=alaymehta1**](https://practice.geeksforgeeks.org/viewSol.php?subId=fe2a679f92c56a79f56603eda13561f7&pid=701374&user=alaymehta1)