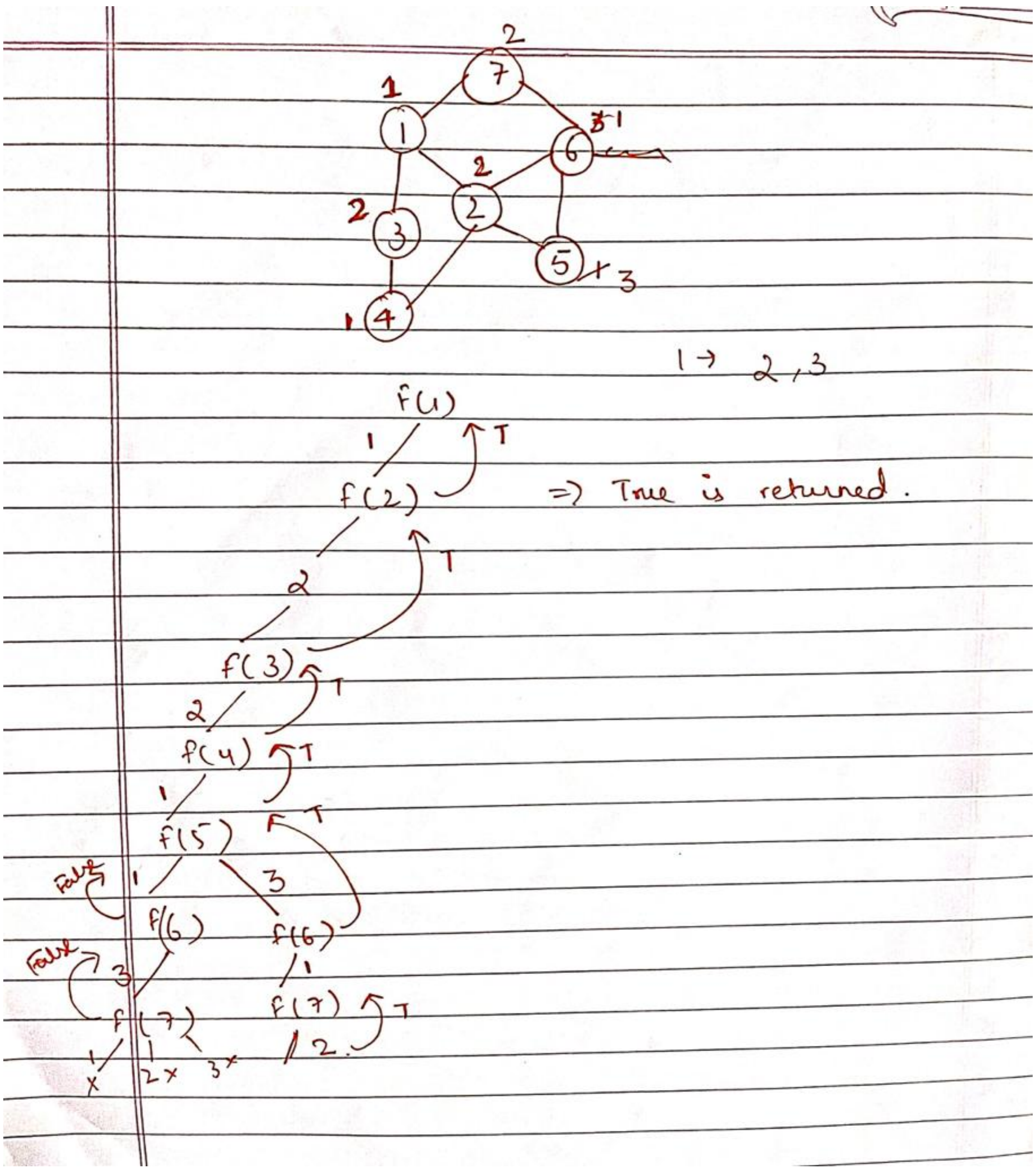
M Coloring Pattern

We try out every color for a node if a color is valid we move onto the next node and find a suitable color we keep a check whether a color can be used or not using function isValid. if it is valid then we allot that color to the specific node using a color array. If any of the adjacent node contains the same color in the color array that color cannot be filled. If all the nodes are traversed successfully that is ind==mat.size() we return true. If a function returns true all its above recursive calls will return true because one possible combination of color is found. If while traversing we found a valid color and keep on calling recursively further nodes but at some point we are left with no color to fill then we backtrack remove the color previously allotted and try out some other color which is valid. If nodes are remaining to be colored but we are left with no color then the loop ends and we simply return a false.



```
#include <bits/stdc++.h>
bool isValid(int ind,int i,vector<int>& color,vector<vector<int>> mat)
{
    for(int j=0;j<mat.size();j++)
    {
        if(mat[ind][j]==1&&color[j]==i)
            return false;
    }
    return true;
}
bool helper(int ind,vector<vector<int>>& mat,vector<int>& color,int m)
{
    if(ind==mat.size())
    {
        return true;
    }
    for(int i=0;i<m;i++)
    {
        if(isValid(ind,i,color,mat))
        {
            color[ind]=i;
            if(helper(ind+1,mat,color,m))
                return true;
            color[ind]=-1;
        }
    }
    return false;
}
string graphColoring(vector<vector<int>> &mat,int m) {
    // Write your code here
    vector<int>  color(mat.size(),-1);
    if(helper(0,mat,color,m))
        return "YES";
    return "NO";

}
```

- Time Complexity : O(N^M)
- Space Complexity : O(N)