Batch: A3	**Roll No.: 16010121045**
Experiment No: 8	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Implementation of Graph Colouring Backtracking Algorithm

Objective: To learn the Backtracking strategy of problem solving for Graph Colouring problem

CO to be achieved:

CO 2	Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies.

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://www.math.utah.edu/~alfeld/queens/queens.html**
4. **http://www-isl.ece.arizona.edu/ece175/assignments275/assignment4a/Solving%208%20queen%20problem.pdf**
5. **http://www.slideshare.net/Tech_MX/8-queens-problem-using-back-tracking**
6. **http://www.mathcs.emory.edu/~cheung/Courses/170.2010/Syllabus/Backtracking/8queens.html**
7. **http://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/**
8. **http://www.hbmeyer.de/backtrack/achtdamen/eight.htm**

Pre Lab/ Prior Concepts:
Data structures, Concepts of algorithm analysis

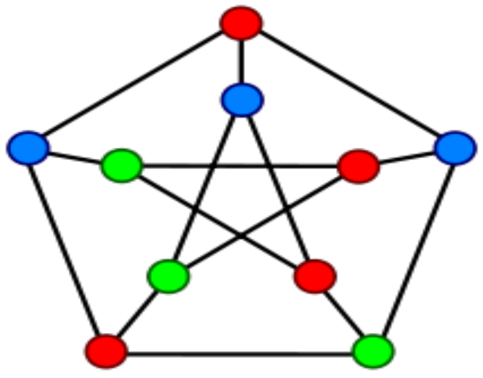
Historical Profile:
Given an undirected graph and a number m, determine if the graph can be colored with at most m colors such that no two adjacent vertices of the graph are colored with same color. Here coloring of a graph means assignment of colors to all vertices.
nput:

1) A 2D array graph [V][V] where V is the number of vertices in graph and graph[V][V] is adjacency matrix representation of the graph.

Output:

An array color [V] that should have numbers from 1 to m. color[i] should represent the color assigned to the ith vertex. The code should also return false if the graph cannot be colored with m colors.

Following is an example graph can be colored with 3 colors.



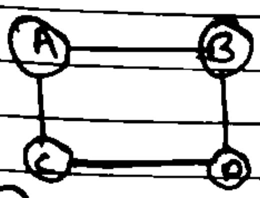
New Concepts to be learned:
Application of algorithmic design strategy to any problem, Backtracking method of problem solving Vs other methods of problem solving problem graph colouring and its applications.

Algorithm Graph colouring Problem:-

```
Algorithm mColoring(k)
// This algorithm was formed using the recursive backtracking
// schema. The graph is represented by its boolean adjacency
// matrix $G[1:n, 1:n]$. All assignments of $1, 2, \ldots, m$ to the
// vertices of the graph such that adjacent vertices are
// assigned distinct integers are printed. k is the index
// of the next vertex to color.
{
    repeat
    {// Generate all legal assignments for $x[k]$.
        NextValue(k); // Assign to $x[k]$ a legal color.
        if ($x[k] = 0$) then return; // No new color possible
        if ($k = n$) then      // At most m colors have been
                               // used to color the n vertices.
            write ($x[1:n]$);
        else mColoring($k + 1$);
    } until (false);
}
```

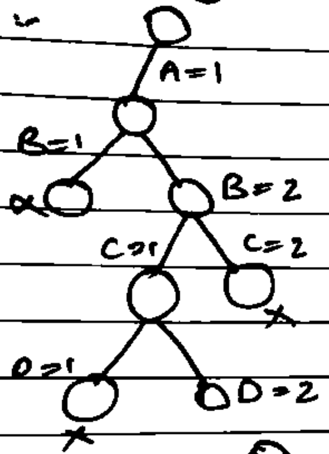
Example Graph Colouring Problem:

Example :-



Find Chromatic number of Input Graph & possible solutions

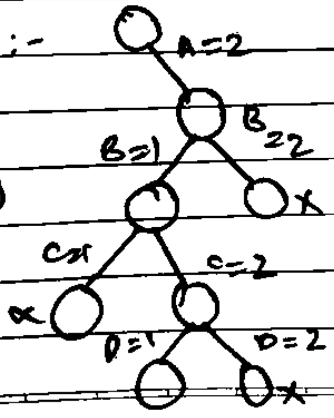
Soln :-



While solving the above tree, first solution comes which says the minimum no. of colours req.

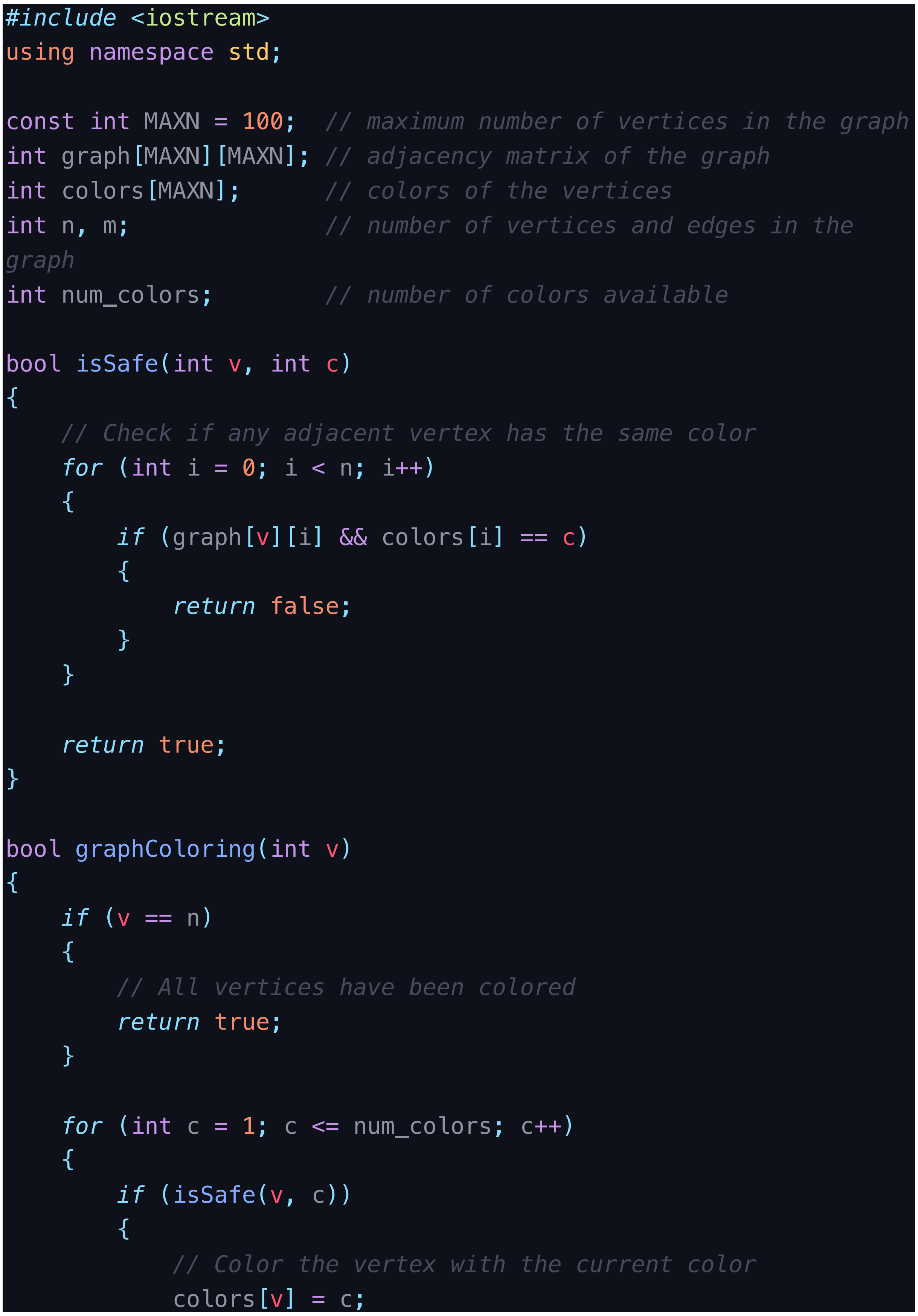
∴ Chromatic num = 2

Other soln :-

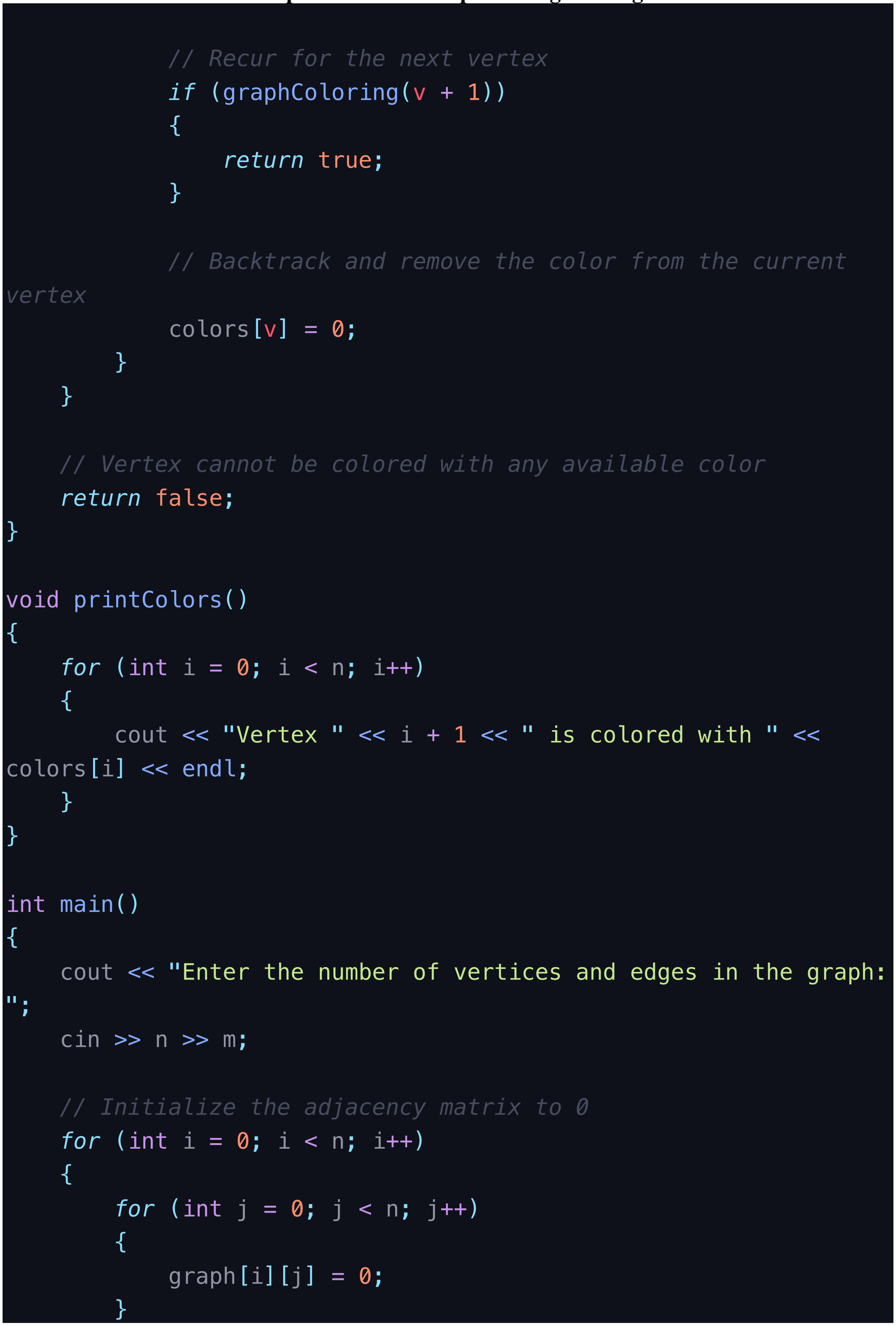


Solutions :-
{1, 2, 1, 2}
{2, 1, 2, 1}

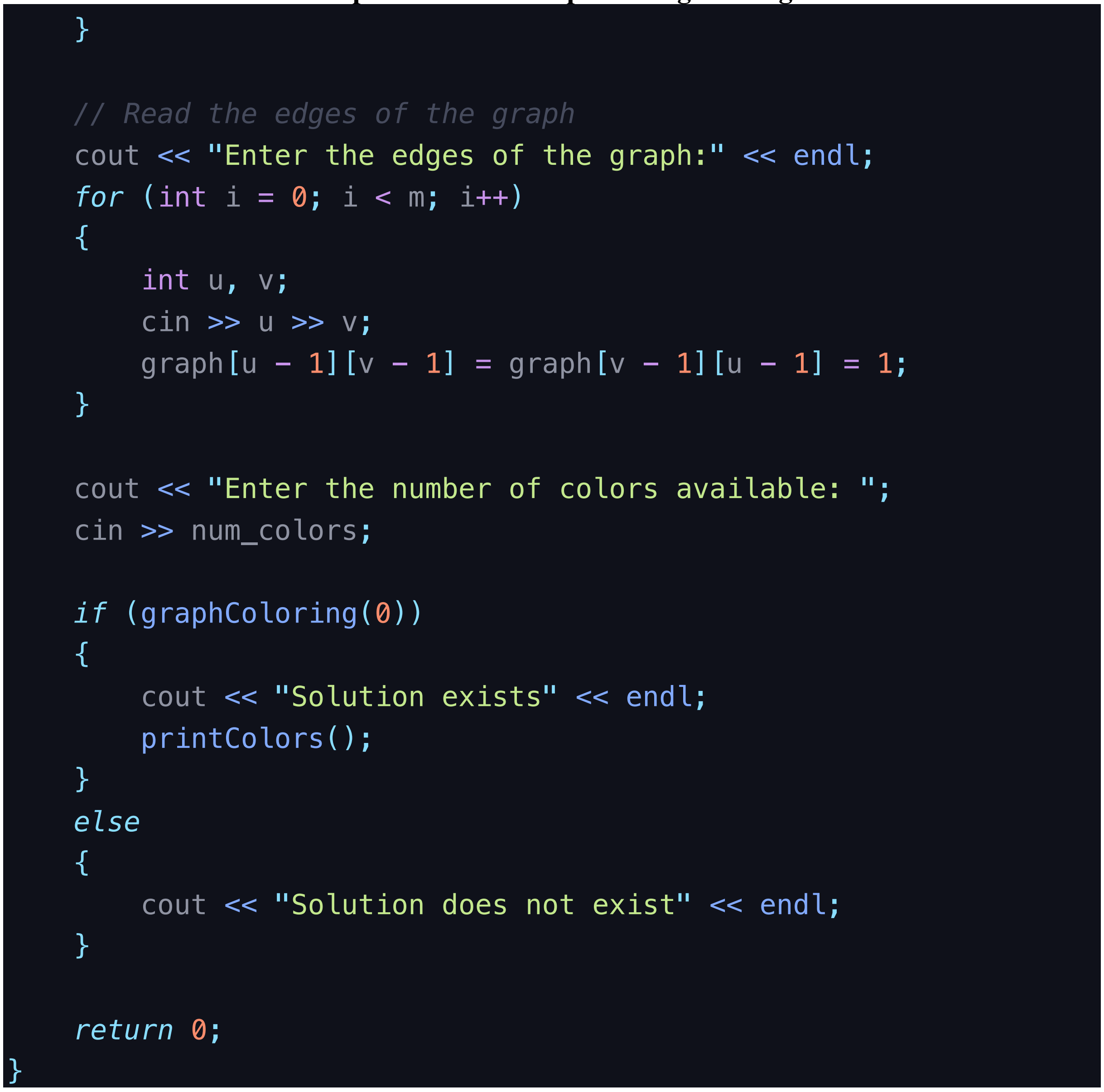
Code:



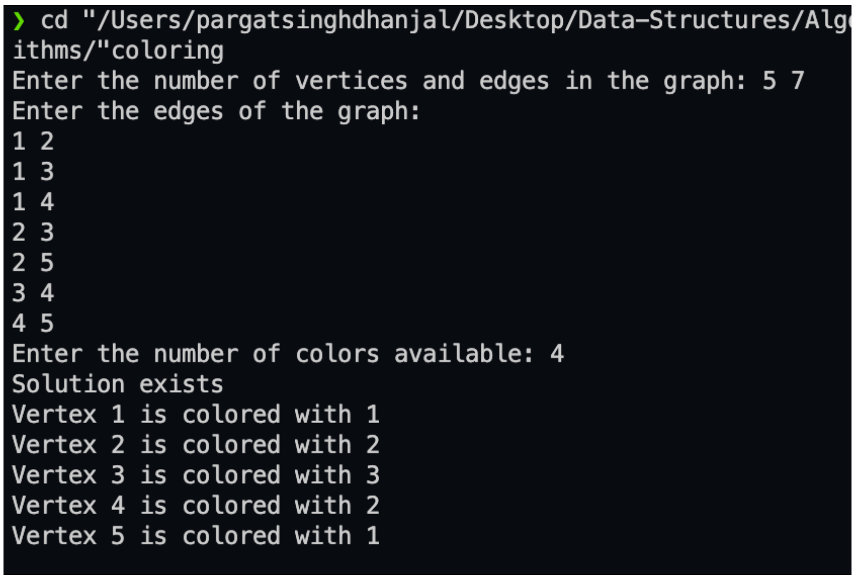
```
#include <iostream>
using namespace std;

const int MAXN = 100;  // maximum number of vertices in the graph
int graph[MAXN][MAXN]; // adjacency matrix of the graph
int colors[MAXN];      // colors of the vertices
int n, m;              // number of vertices and edges in the
graph
int num_colors;        // number of colors available

bool isSafe(int v, int c)
{
    // Check if any adjacent vertex has the same color
    for (int i = 0; i < n; i++)
    {
        if (graph[v][i] && colors[i] == c)
        {
            return false;
        }
    }

    return true;
}

bool graphColoring(int v)
{
    if (v == n)
    {
        // All vertices have been colored
        return true;
    }

    for (int c = 1; c <= num_colors; c++)
    {
        if (isSafe(v, c))
        {
            // Color the vertex with the current color
            colors[v] = c;
```



```
            // Recur for the next vertex
            if (graphColoring(v + 1))
            {
                return true;
            }

            // Backtrack and remove the color from the current
vertex
            colors[v] = 0;
        }
    }

    // Vertex cannot be colored with any available color
    return false;
}

void printColors()
{
    for (int i = 0; i < n; i++)
    {
        cout << "Vertex " << i + 1 << " is colored with " <<
colors[i] << endl;
    }
}

int main()
{
    cout << "Enter the number of vertices and edges in the graph:
";
    cin >> n >> m;

    // Initialize the adjacency matrix to 0
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            graph[i][j] = 0;
        }
```



```
    }

    // Read the edges of the graph
    cout << "Enter the edges of the graph:" << endl;
    for (int i = 0; i < m; i++)
    {
        int u, v;
        cin >> u >> v;
        graph[u - 1][v - 1] = graph[v - 1][u - 1] = 1;
    }

    cout << "Enter the number of colors available: ";
    cin >> num_colors;

    if (graphColoring(0))
    {
        cout << "Solution exists" << endl;
        printColors();
    }
    else
    {
        cout << "Solution does not exist" << endl;
    }

    return 0;
}
```



```
❯ cd "/Users/pargatsinghdhanjal/Desktop/Data-Structures/Alg
ithms/"coloring
Enter the number of vertices and edges in the graph: 5 7
Enter the edges of the graph:
1 2
1 3
1 4
2 3
2 5
3 4
4 5
Enter the number of colors available: 4
Solution exists
Vertex 1 is colored with 1
Vertex 2 is colored with 2
Vertex 3 is colored with 3
Vertex 4 is colored with 2
Vertex 5 is colored with 1
```

Analysis of Backtracking solution for Graph Colouring Problem:

In the backtracking approach of the graph colouring problem, we are generating total m^V combinations of the colour. So, it also requires exponential time.

Time Complexity: In the backtracking approach of the graph colouring problem, the time complexity is $O(m^V)$. In the backtracking approach, the average time complexity is less than $O(m^V)$.

Space Complexity: In the backtracking approach to the graph colouring problem, we are not using any extra space but we are using the recursive stack for the recursive function call. So, the space complexity is O(V).

Conclusion:

We learnt the Backtracking strategy of problem solving for Graph Colouring problem.