PART A

(PART A: TO BE REFFERED BY STUDENTS)

Experiment No. 08

A.1 Aim:

Write a program to implement graph coloring using Backtracking technique.

A.2 Prerequisite:

A.3 Outcome:

After successful completion of this experiment, students will be able to evaluate optimal solution using backtracking and branch-and-bound formulation to deal with hard problems.

A.4 Theory:

Backtracking is an algorithmic paradigm that tries different solutions until finds a solution that "works". Problems which are typically solved using backtracking technique have following property in common. These problems can only be solved by trying every possible configuration and each configuration is tried only once.

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.

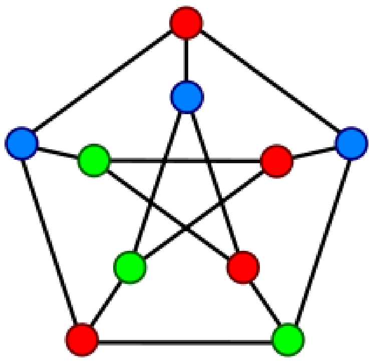
A 2D array graph[V][V] where V is the number of vertices in graph and graph[V][V] is adjacency matrix representation of the graph is input to the algorithm. A value graph[i][j] is 1 if there is a direct edge from i to j, otherwise graph[i][j] is 0. An integer m which is maximum number of colors that can be used.

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

Algorithm:

```
If all colors are assigned,
    print vertex assigned colors
Else
    a. Trying all possible colors, assign a color to the vertex
    b. If color assignment is possible, recursivelty assign colors to next vertices
    c. If color assignment is not possible, de-assign color, return False
```

Example:



Time Complexity:

Time complexity of graph coloring problem is O(nm^n) where n=no. of vertex, m=no. of color used.

PART B

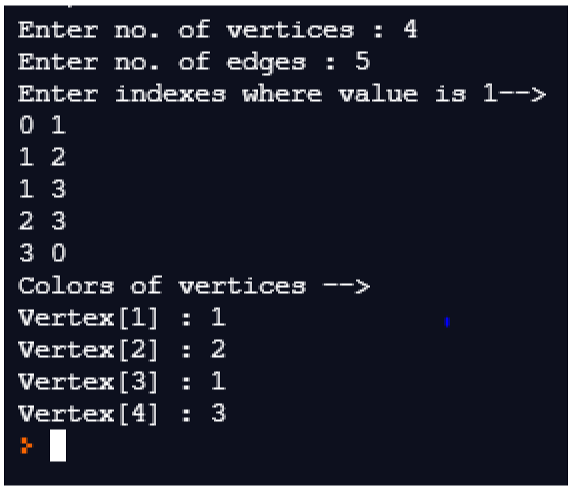
Roll No.: B-81	Name: Bhushan Prashant Ghevde
Class: SE-B	Batch: B4
Date of Experiment: 06/05/2021	Date of Submission: 08/05/2021
Grade:	

B.1 Software Code written by student:

(Paste your code completed during the 2 hours of practical in the lab here)

```
#include<stdio.h>
int G[50][50],x[50];  //G:adjacency matrix,x:colors
void next_color(int k){
   int i,j;
   x[k]=1;  //coloring vertex with color1
   for(i=0;i<k;i++){ //checking all k-1 vertices-backtracking
     if(G[i][k]!=0 && x[k]==x[i])  //if connected and has same color
       x[k]=x[i]+1;  //assign higher color than x[i]
   }
}

int main(){
  int n,e,i,j,k,l;
  printf("Enter no. of vertices : ");
  scanf("%d",&n);  //total vertices
  printf("Enter no. of edges : ");
  scanf("%d",&e);  //total edges

  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
      G[i][j]=0;  //assign 0 to all index of adjacency matrix

  printf("Enter indexes where value is 1-->\n");
  for(i=0;i<e;i++){
    scanf("%d %d",&k,&l);
    G[k][l]=1;
    G[l][k]=1;
  }

  for(i=0;i<n;i++)
    next_color(i);  //coloring each vertex

  printf("Colors of vertices -->\n");
  for(i=0;i<n;i++)  //displaying color of each vertex
    printf("Vertex[%d] : %d\n",i+1,x[i]);

  return 0;
}
```

B.2 Input and Output:


```
Enter no. of vertices : 4
Enter no. of edges : 5
Enter indexes where value is 1-->
0 1
1 2
1 3
2 3
3 0
Colors of vertices -->
Vertex[1] : 1
Vertex[2] : 2
Vertex[3] : 1
Vertex[4] : 3
> 
```

B.3 Observations and learning:

(Students are expected to comment on the output obtained with clear observations and learning for each task/ sub part assigned)

In this problem, for any given graph G we will have to color each of the vertices in G in such a way that no two adjacent vertices get the same color and the least number of colors are used.

B.4 Conclusion:

(Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.3)

In this experiment we have studied a program to implement graph coloring using Backtracking technique.

B.5 Question of Curiosity

(To be answered by student based on the practical performed and learning/observations)

Q.1: What are applications of backtracking?

Answer :

1. To find all Hamiltonian Paths present in a graph.

2. To solve the N Queen problem.

3. Maze solving problem.

4. The Knight's tour problem.

Q.2: What are different application graph coloring problem?

Answer :

1) Making Schedule or Time Table: Suppose we want to make am exam schedule for a university. We have list different subjects and students enrolled in every subject. Many subjects would have common students (of same batch, some backlog students, etc). How do we schedule the exam so that no two exams with a common student are scheduled at same time? How many minimum time slots are needed to schedule all exams? This problem can be represented as a graph where every vertex is a subject and an edge between two vertices mean there is a common student. So this is a graph coloring problem where minimum number of time slots is equal to the chromatic number of the graph.
2) Mobile Radio Frequency Assignment: When frequencies are assigned to towers, frequencies assigned to all towers at the same location must be different. How to assign frequencies with this constraint? What is the minimum number of frequencies needed? This problem is also an instance of graph coloring problem where every tower represents a vertex and an edge between two towers represents that they are in range of each other.
3) Sudoku: Sudoku is also a variation of Graph coloring problem where every cell represents a vertex. There is an edge between two vertices if they are in same row or same column or same block.
4) Register Allocation: In compiler optimization, register allocation is the process of assigning a large number of target program variables onto a small number of CPU registers. This problem is also a graph coloring problem.
5) Bipartite Graphs: We can check if a graph is Bipartite or not by coloring the graph using two colors. If a given graph is 2-colorable, then it is Bipartite, otherwise not.
See this for more details.
6) Map Coloring: Geographical maps of countries or states where no two adjacent cities cannot be assigned same color. Four colors are sufficient to color any map (See Four Color Theorem)

Q.3: What is difference between backtracking and branch and bound techniques?

Answer :

Parameter	**Backtracking**	**Branch and Bound**
Approach	Backtracking is used to find all possible solutions available to a problem. When it realises that it has made a bad choice, it undoes the last choice by backing it up. It searches the state space tree until it has found a solution for the problem.	Branch-and-Bound is used to solve optimisation problems. When it realises that it already has a better optimal solution that the pre-solution leads to, it abandons that pre-solution. It completely searches the state space tree to get optimal solution.

Traversal	Backtracking traverses the state space tree by DFS(Depth First Search) manner.	Branch-and-Bound traverse the tree in any manner, DFS or BFS.
Function	Backtracking involves feasibility function.	Branch-and-Bound involves a bounding function.
Problems	Backtracking is used for solving Decision Problem.	Branch-and-Bound is used for solving Optimisation Problem.
Searching	In backtracking, the state space tree is searched until the solution is obtained.	In Branch-and-Bound as the optimum solution may be present any where in the state space tree, so the tree need to be searched completely.
Efficiency	Backtracking is more efficient.	Branch-and-Bound is less efficient.
Applications	Useful in solving N-Queen Problem, Sum of subset.	Useful in solving Knapsack Problem, Travelling Salesman Problem.

Q.4: Which technique id more efficient? Backtracking or branch and bound techniques?

Answer :

Branch-and-Bound is used for solving Optimisation Problem. In backtracking, the state space tree is searched until the solution is obtained. In Branch-and-Bound as the optimum solution may be present any where in the state space tree, so the tree need to be searched completely. Backtracking is more efficient.

****