Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year:2025), B.Sc. in CSE (Day)

Lab Report NO: 03

Course Title: Artificial Intelligence Lab

Course Code: CSE 316 Section: 223-D4

Lab Experiment Name: GraphColoring

Student Details

	Name	ID
1.	Ruhul Amin	223002097

Lab Date :

Submission Date : 29-12-25

Course Teacher's Name : Sabbir Hosen Mamun

Lab Report Status

Marks: ..... Signature:.....

Comments:..... Date:.....

- **TITLE OF THE LAB REPORT EXPERIMENT:**

You are given an undirected graph with N vertices and M edges. Your task is to implement a Python program that uses Backtracking to determine whether the graph can be colored using K colors such that no two adjacent vertices share the same color. The input will be read from a file, where the first line contains three integers: N (number of vertices, numbered from 0 to N−1), M (number of edges), and K (number of available colors). Each of the following M lines contains two integers u and v, representing an undirected edge between vertex u and vertex v.

- **OBJECTIVES:**

1. To understand the **Graph Coloring problem** in Artificial Intelligence.
2. To apply the **Backtracking algorithm** to solve a constraint satisfaction problem.
3. To determine whether a graph can be colored using **K colors** without conflict.
4. To implement the solution using **Python programming language**.
5. To analyze the feasibility of coloring for different values of K.

- **PROCEDURE:**

a) Read the number of vertices (N), edges (M), and available colors (K) from the input file.
b) Construct an adjacency list to represent the undirected graph.
c) Initialize a color array where all vertices are initially uncolored.
d) Start coloring vertices one by one using recursion.
e) For each vertex, try assigning colors from 1 to K.
f) Check whether the selected color is safe by ensuring no adjacent vertex has the same color.
g) If a conflict occurs, backtrack and try a different color.
h) Continue until all vertices are colored or no valid coloring is possible.
i) Display the result and color assignment if successful.

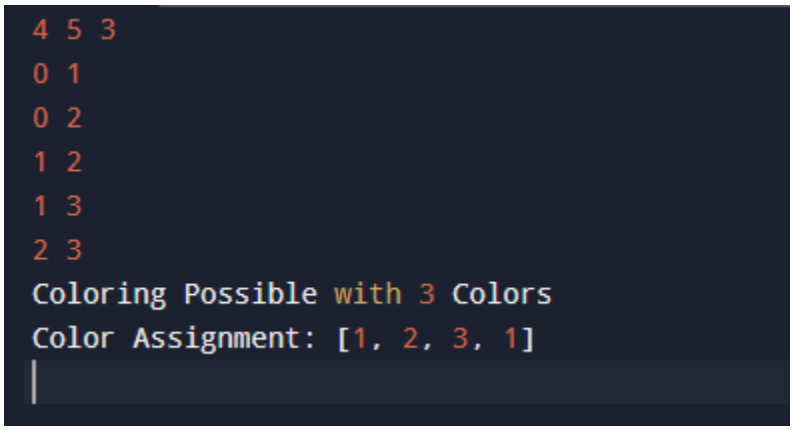
- **IMPLEMENTATION AND OUTPUT:**

CODE:

```
def is_safe(vertex, graph, colors, color):
    for neighbor in graph[vertex]:
        if colors[neighbor] == color:
            return False
    return True


def graph_coloring_util(vertex, graph, colors, N, K):
    if vertex == N:
        return True

    for color in range(1, K + 1):
        if is_safe(vertex, graph, colors, color):
            colors[vertex] = color
            if graph_coloring_util(vertex + 1, graph, colors, N, K):
                return True
            colors[vertex] = 0  # Backtrack

    return False


def graph_coloring(N, graph, K):
    colors = [0] * N
    if graph_coloring_util(0, graph, colors, N, K):
        return True, colors
    return False, None


# -------- MAIN --------
with open("input.txt", "r") as f:
    data = list(map(int, f.read().split()))

idx = 0
N, M, K = data[idx], data[idx+1], data[idx+2]
idx += 3

graph = {i: [] for i in range(N)}

for _ in range(M):
    u, v = data[idx], data[idx+1]
    idx += 2
    graph[u].append(v)
    graph[v].append(u)
```

```
possible, assignment = graph_coloring(N, graph, K)

if possible:
    print(f"Coloring Possible with {K} Colors")
    print(f"Color Assignment: {assignment}")
else:
    print(f"Coloring Not Possible with {K} Colors")
```

OUTPUT:



5. ANALYSIS AND DISCUSSION

The backtracking-based graph coloring algorithm efficiently explores possible color assignments while ensuring that no adjacent vertices share the same color. When sufficient colors are available, the algorithm successfully finds a valid coloring. In contrast, when the number of colors is insufficient, it correctly identifies that no solution exists. Although effective for small and medium-sized graphs, the algorithm's time complexity increases exponentially with the number of vertices, making it less suitable for very large graphs.

6. SUMMARY

This experiment demonstrated the use of the **Backtracking algorithm** to solve the Graph Coloring problem. The algorithm systematically assigns colors while satisfying adjacency constraints and correctly determines whether a valid coloring is possible for a given number of colors. The results confirm that backtracking is a reliable approach for solving constraint satisfaction problems in artificial intelligence.

Github link: https://github.com/ruhulaminn1316/A-Artificial-Intelligence-Lab