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 404-CSE(181) **Section:** 223_D4

Lab Experiment Name: Implementation of Graph Coloring Problem Using Backtracking Algorithm

Student Details

Name	ID
Mujahidul Islam	193002052

Lab Date: 29/11/2025

Submission Date: 19/12/2025

Course Teacher's Name: Md. Sabbir Hosen Mamun

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: …… Signature:.....

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

1. Introduction

Graph coloring is a classic problem in computer science where colors are assigned to graph vertices so that no two adjacent vertices have the same color. It has practical applications in scheduling, map coloring, and network resource allocation. In this experiment, the Graph Coloring Problem is solved using the Backtracking algorithm, which systematically explores valid color assignments while avoiding conflicts.

2. Objectives

- To understand the Graph Coloring Problem.
- To implement the Backtracking algorithm in Python.
- To determine whether a graph can be colored using a fixed number of colors.
- To gain experience with graph-based constraint satisfaction problems.

3. Implementation

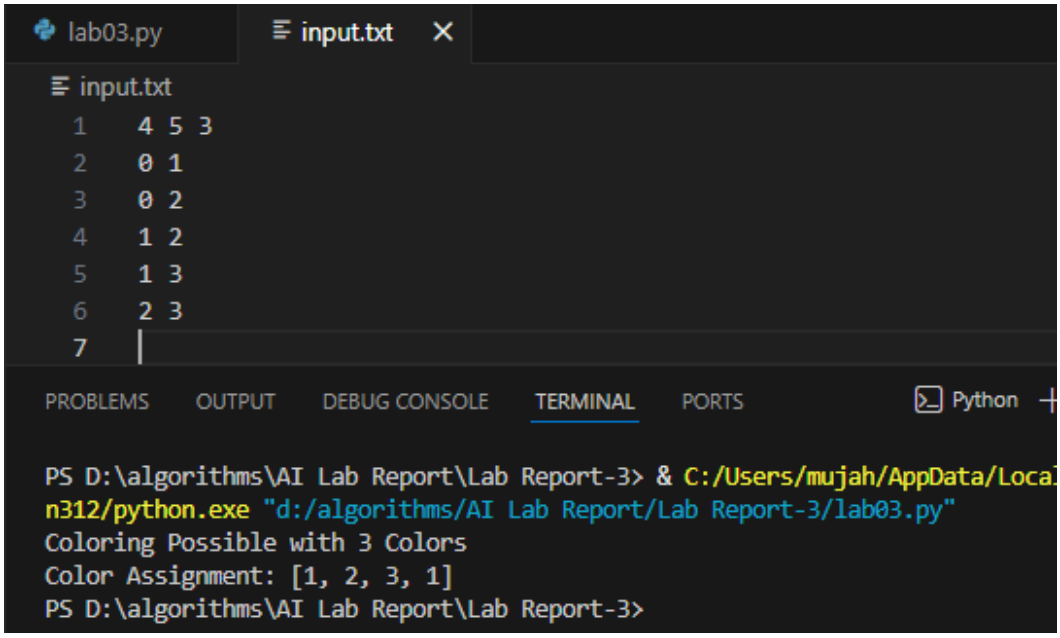
Welcome | lab03.py | input.txt

lab03.py > read_input

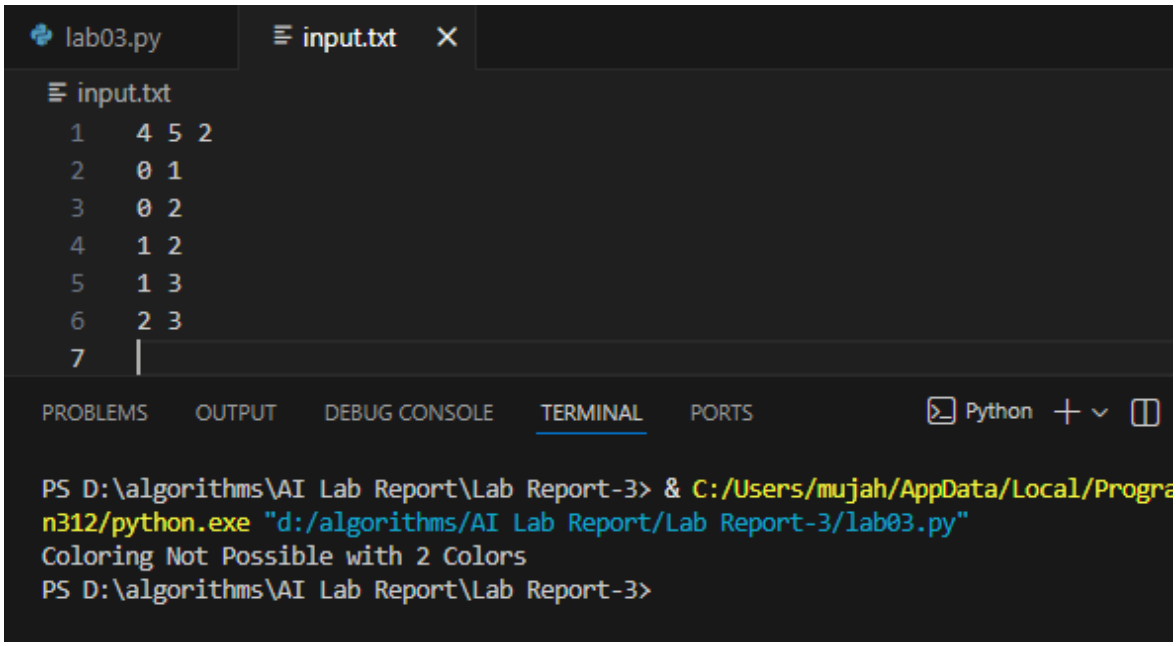
```python
def is_safe(v, graph, colors, c):
    for neighbor in graph[v]:
        if colors[neighbor] == c:
            return False
    return True
def graph_coloring_util(graph, k, colors, v, n):
    if v == n:
        return True
    for c in range(1, k + 1):
        if is_safe(v, graph, colors, c):
            colors[v] = c
            if graph_coloring_util(graph, k, colors, v + 1, n):
                return True
            colors[v] = 0  # Backtrack
    return False
def graph_coloring(n, graph, k):
    colors = [0] * n
    if graph_coloring_util(graph, k, colors, 0, n):
        print(f"Coloring Possible with {k} Colors")
        print("Color Assignment:", colors)
    else:
        print(f"Coloring Not Possible with {k} Colors")
def read_input(filename):
    with open(filename, 'r') as file:
        n, m, k = map(int, file.readline().split())
        graph = {i: [] for i in range(n)}
        for _ in range(m):
            u, v = map(int, file.readline().split())
            graph[u].append(v)
            graph[v].append(u)
    return n, graph, k
if __name__ == "__main__":
```

4. Output

Case-01:



Case-02:



5. Discussion

The graph is represented using an adjacency list, and colors are assigned recursively to each vertex. Before assigning a color, the algorithm checks whether the color is safe by ensuring that adjacent vertices do not share the same color. If a conflict occurs, the algorithm backtracks and tries another color. The program successfully identifies both possible and impossible coloring cases, proving the correctness of the approach.

6. Conclusion

The experiment successfully implemented the Graph Coloring Problem using the Backtracking algorithm. It demonstrated how backtracking efficiently handles constraints and ensures correct coloring. Although the algorithm has high time complexity, it is effective for small graphs and educational purposes.