



Green University of Bangladesh

*Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2025), B.Sc. in CSE (Day)*

Graph Coloring using Backtracking

*Course Title: Artificial Intelligence Lab
Course Code: CSE-316
Section: 221-D22*

Students Details

Name	ID
Md Showaib Rahman Tanveer	221902084

*Submission Date: 03/04/2025
Course Teacher's Name: Md. Sabbir Hosen Mamun*

[For teachers use only: **Don't write anything inside this box**]

<u>Lab Report Status</u>	
Marks:	Signature:
Comments:	Date:

1 Introduction

Graph coloring is an important problem in graph theory and combinatorial optimization. It involves assigning colors to vertices of a graph such that no two adjacent vertices share the same color. This report presents an implementation of graph coloring using a backtracking approach.

2 Objectives

- Understand the graph coloring problem.
- Implement a backtracking solution to assign colors to vertices.
- Determine whether a valid coloring exists for a given number of colors.

3 Procedure

1. Accept user input for the number of vertices, edges, and available colors.
2. Construct the adjacency matrix representation of the graph.
3. Use a backtracking approach to attempt coloring the graph.
4. Display the color assignment if a valid coloring is found.
5. Indicate failure if the graph cannot be colored with the given colors.

4 Code

```
def is_safe(graph, vertex, colors, color, V):
    for i in range(V):
        if graph[vertex][i] == 1 and colors[i] == color:
            return False
    return True

def graph_coloring_util(graph, k, colors, vertex, V):
    if vertex == V:
        return True

    for color in range(1, k + 1):
        if is_safe(graph, vertex, colors, color, V):
            colors[vertex] = color
            if graph_coloring_util(graph, k, colors, vertex + 1, V):
                return True
            colors[vertex] = 0
    return False
```

```

def solve_graph_coloring():
    print("Enter the number of vertices (N), edges (M), and colors (K):")
    N, M, K = map(int, input().split())

    graph = [[0] * N for _ in range(N)]

    print("\nEnter the edges (u v):")
    for _ in range(M):
        u, v = map(int, input().split())
        graph[u][v] = 1
        graph[v][u] = 1

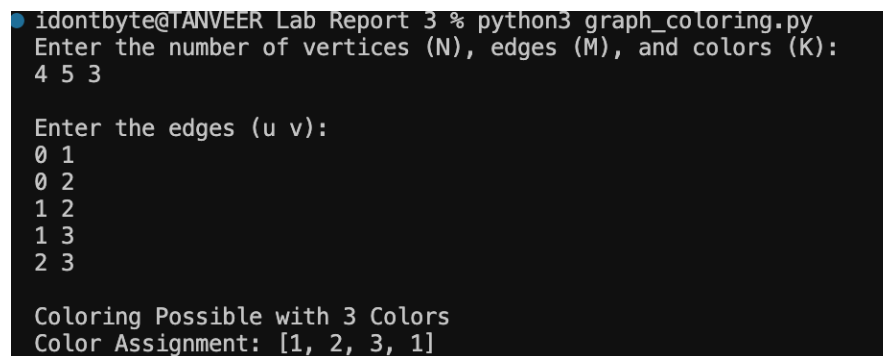
    colors = [0] * N

    if graph_coloring_util(graph, K, colors, 0, N):
        print(f"\nColoring Possible with {K} Colors")
        print(f"Color Assignment: {colors}")
    else:
        print(f"\nColoring Not Possible with {K} Colors")

if __name__ == "__main__":
    solve_graph_coloring()

```

5 Output



```

identbyte@TANVEER Lab Report 3 % python3 graph_coloring.py
Enter the number of vertices (N), edges (M), and colors (K):
4 5 3

Enter the edges (u v):
0 1
0 2
1 2
1 3
2 3

Coloring Possible with 3 Colors
Color Assignment: [1, 2, 3, 1]

```

Figure 1: Example Output of the Graph Coloring Algorithm

6 Conclusion

The graph coloring problem is effectively solved using backtracking. The algorithm successfully determines whether a valid coloring is possible for a given number of colors. The results demonstrate the efficiency of backtracking in solving constraint satisfaction problems.

7 GitHub Repository

The complete implementation can be found in the following repository:

<https://github.com/idontbyte69/Academic/tree/master/AI%20LAB%20CSE%20316/Lab%20Report/Lab%20Report%203>