Vidyavardhini's College of Engineering & Technology

Department of Computer Science and Engineering (Data Science)

ACADEMIC YEAR: 2024-25

Course: Analysis of Algorithm Lab

Course code: CSL401

Year/Sem: SE/IV

Experiment No.: 09
Aim: To implement graph coloring problem using backtracking technique
Name: SOHAM HEMENDRA RAUT
Roll Number: 24
Date of Performance: 27/03/2025
Date of Submission: 03/04/2025

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission.	10	
Total	**20**	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	5	3	2
Understanding	5	3	2
Journal work and timely submission.	10	8	4

Checked by

Name of Faculty : Mrs. Komal Champanerkar

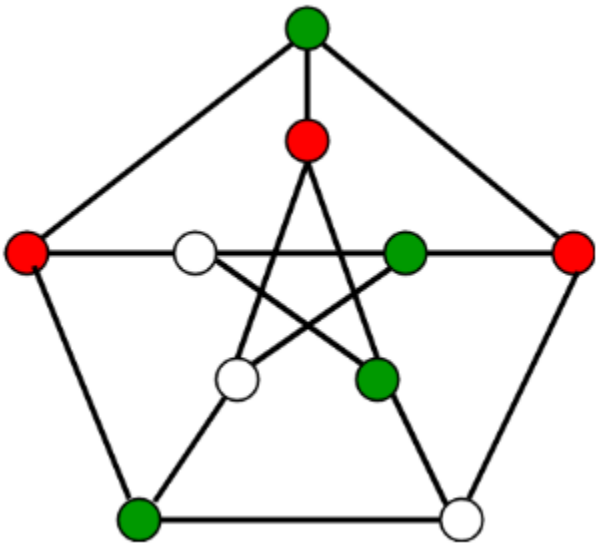
Signature :

Date :

- **Aim: To implement graph coloring problem using backtracking technique**
- **Theory:**

We are given a graph, we need to assign colors to the vertices of the graph. In the graph coloring problem, we have a graph and m colors, we need to find a way to color the vertices of the graph using the m colors such that any two adjacent vertices are not having the same color.

The backtracking approach to solving the graph coloring problem can be to assign the colors one after the other to the various vertices. The coloring will start with the first index only but before assigning any color, we would first check if it satisfies the constraint or not (i.e. no two adjacent vertices have the same color). If the current color assignment does not violate the condition then add it into the solution else, backtrack by returning false.



- **Algorithm:**

Step 1: Select any node of a given graph and assign it the color number 1.

Step 2: Then select remaining nodes of a graph one by one and assign the higher numbered color to the from the list of available colors

Step 3: For each color assignment, apply the bounding function to check whether two

adjacent nodes have different colors

Step 4: If the partial solution is infeasible, discard that path without further exploration and backtrack to the previous level and assign another valid color to the node.

Step 5: Repeat step 2 to 5, until the complete solution is found.

Step 6: IF not fond, report.

Step 7: Exit.

❖ **Program:**

```
def is_safe(node, color, graph, colors, c):
    for neighbor in range(len(graph)):
        if graph[node][neighbor] == 1 and colors[neighbor] == c:
            return False
    return True

def solve(node, graph, m, colors):
    if node == len(graph):
        return True

    for c in range(1, m + 1):
        if is_safe(node, color=c, graph=graph, colors=colors, c=c):
            colors[node] = c
            if solve(node + 1, graph, m, colors):
                return True
            colors[node] = 0  # Backtrack
    return False

def graph_coloring(graph, m):
    n = len(graph)
    colors = [0] * n
```

```
    if solve(0, graph, m, colors):
        print("Solution Exists: Following are the assigned colors:")
        print(colors)
    else:
        print("No solution exists.")


graph = [
    [0, 1, 1, 1],
    [1, 0, 1, 0],
    [1, 1, 0, 1],
    [1, 0, 1, 0]
]

m = 3  # Number of colors
graph_coloring(graph, m)
```

❖ **Output:**

```
Solution Exists: Following are the assigned colors:
[1, 2, 3, 2]
```

❖ **Conclusion:**

The Graph Coloring Problem is a well-known example of a constraint satisfaction problem that can be effectively tackled using the backtracking technique. This method involves exploring all possible color combinations and reverting when conflicts arise, ensuring that no two adjacent vertices are assigned the same color. While this approach can be time-consuming for larger graphs, it remains a straightforward and reliable method for smaller instances. Additionally, it provides a foundation for more advanced optimization techniques, such as greedy algorithms and graph heuristics.