DAA PRACTICAL 8

Name -Rishabh Jain

Branch - CSE A

Roll No- 54

Aim - Implement Graph Colouring algorithm use Graph colouring concept.

Problem Statement - A GSM is a cellular network with its entire geographical range divided into hexadecimal cells. Each cell has a communication tower which connects with mobile phones within cell. Assume this GSM network operates in different frequency ranges. Allot frequencies to each cell such that no adjacent cells have same frequency range. Consider an undirected graph G = (V, E) shown in fig. Find the colour assigned to each node using Backtracking method. Input is the adjacency matrix of a graph G(V, E), where V is the number of Vertices and E is the number of edges.

CODE -

```
def main():
    graph = [[0 if i == j else 1 for j in range(5)] for i in range(5)]

    print("The given input matrix:")
    for row in graph:
        print(" ".join(map(str, row)))

    print()

    colored = [0] * len(graph)
    all_possible = []
    m_color(all_possible, graph, colored, 0)

    print("(Index indicates vertex number)\n\n")
    print("All possible color combination sets:")
    for coloring in all_possible:
        print(coloring)
    print("The Total solutions for the graphs are: " + str(len(all_possible)))


def m_color(set, matrix, colored, i):
    while True:
        next_color(i, colored, matrix)
        if colored[i] == 0:
            return
        if i == len(colored) - 1:
            set.append(colored.copy())
        else:
            m_color(set, matrix, colored, i + 1)


def next_color(i, colored, matrix):
    while True:
        j = 0
        next_col = (colored[i] + 1) % (len(colored) + 1)
        colored[i] = next_col
        if next_col == 0:
            return
        for j in range(len(matrix)):
            if matrix[i][j] == 1 and colored[j] == next_col:
                break
        if j == len(matrix) - 1:
            return


if __name__ == "__main__":
    main()
```

```
The given input matrix:
0 1 1 1 1
1 0 1 1 1
```

```
1 1 0 1 1
1 1 1 0 1
1 1 1 1 0

(Index indicates vertex number)


All possible color combination sets:
[1, 2, 3, 4, 5]
[1, 2, 3, 5, 4]
[1, 2, 4, 3, 5]
[1, 2, 4, 5, 3]
[1, 2, 5, 3, 4]
[1, 2, 5, 4, 3]
[1, 3, 2, 4, 5]
[1, 3, 2, 5, 4]
[1, 3, 4, 2, 5]
[1, 3, 4, 5, 2]
[1, 3, 5, 2, 4]
[1, 3, 5, 4, 2]
[1, 4, 2, 3, 5]
[1, 4, 2, 5, 3]
[1, 4, 3, 2, 5]
[1, 4, 3, 5, 2]
[1, 4, 5, 2, 3]
[1, 4, 5, 3, 2]
[1, 5, 2, 3, 4]
[1, 5, 2, 4, 3]
[1, 5, 3, 2, 4]
[1, 5, 3, 4, 2]
[1, 5, 4, 2, 3]
[1, 5, 4, 3, 2]
[2, 1, 3, 4, 5]
[2, 1, 3, 5, 4]
[2, 1, 4, 3, 5]
[2, 1, 4, 5, 3]
[2, 1, 5, 3, 4]
[2, 1, 5, 4, 3]
[2, 3, 1, 4, 5]
[2, 3, 1, 5, 4]
[2, 3, 4, 1, 5]
[2, 3, 4, 5, 1]
[2, 3, 5, 1, 4]
[2, 3, 5, 4, 1]
[2, 4, 1, 3, 5]
[2, 4, 1, 5, 3]
[2, 4, 3, 1, 5]
[2, 4, 3, 5, 1]
[2, 4, 5, 1, 3]
[2, 4, 5, 3, 1]
[2, 5, 1, 3, 4]
[2, 5, 1, 4, 3]
[2, 5, 3, 1, 4]
[2, 5, 3, 4, 1]
[2, 5, 4, 1, 3]
```