Name: Avaneesh Pillay Roll No:45_B3

Practical 8

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

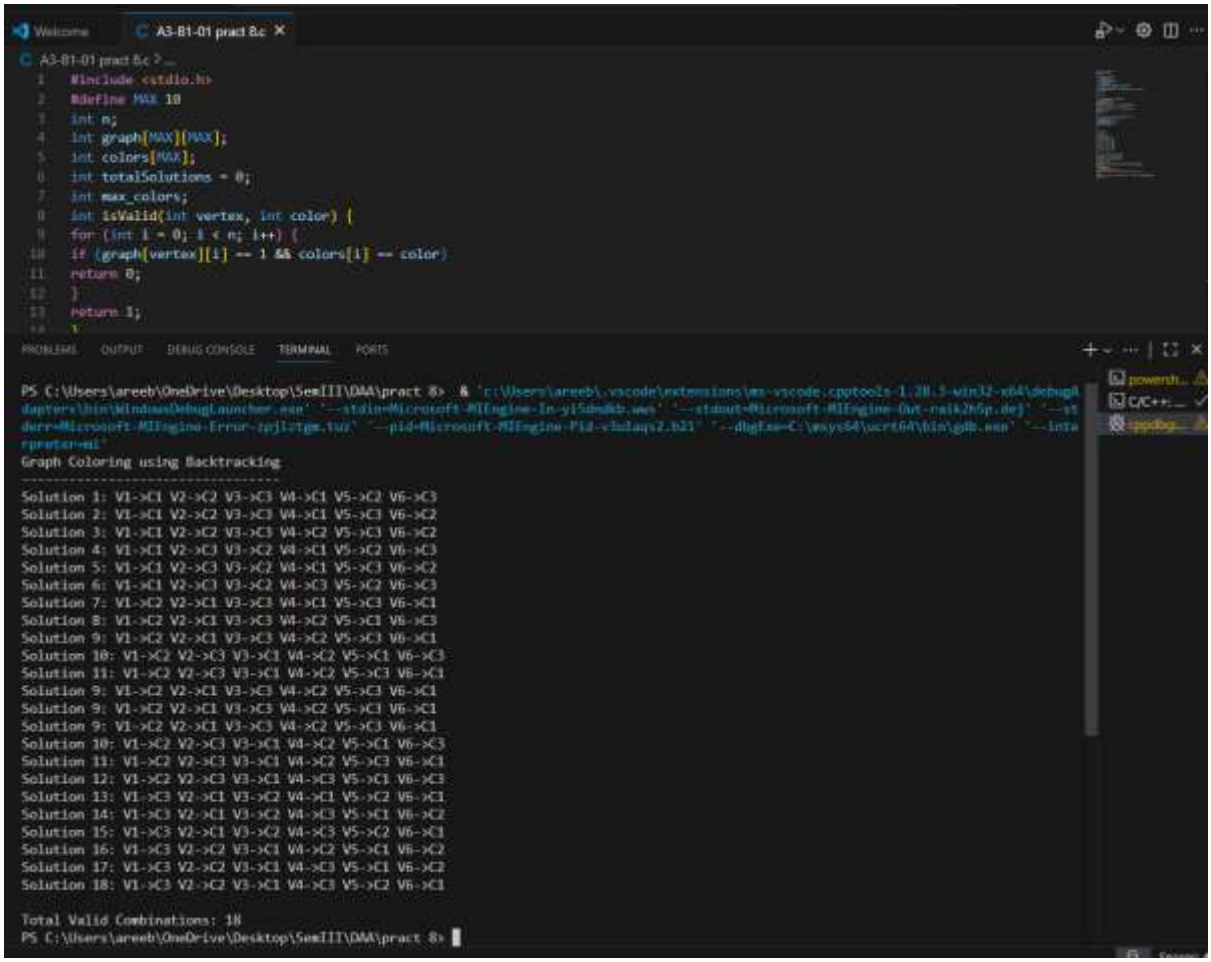
number of Vertices and E is the number of edges..

Code:

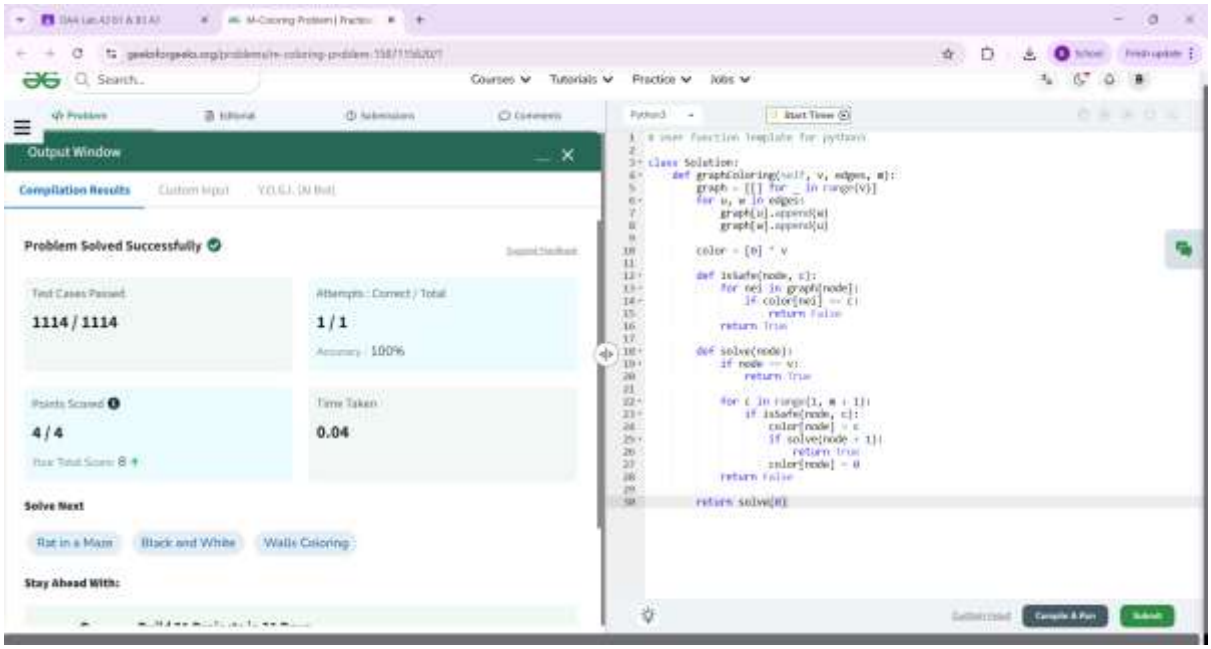
```
#include <stdio.h>
#define MAX 10
int n;
int graph[MAX][MAX];
int colors[MAX];
int totalSolutions = 0;
int max_colors;
int isValid(int vertex, int color) {
for (int i = 0; i < n; i++) {
if (graph[vertex][i] == 1 && colors[i] == color)
return 0;
}
return 1;
}
void colorGraph(int vertex) {
if (vertex == n) {
totalSolutions++;
printf("Solution %d: ", totalSolutions);
for (int i = 0; i < n; i++)
printf("V%d->C%d ", i + 1, colors[i]);
printf("\n");
return;
```

```
}
for (int c = 1; c <= max_colors; c++) {
if (isValid(vertex, c)) {
colors[vertex] = c;
colorGraph(vertex + 1);
colors[vertex] = 0;
}
}
}
int main() {
n = 6;
max_colors = 3;
int temp[6][6] = {
{0, 1, 1, 0, 1, 1},
{1, 0, 1, 0, 0, 0},
{1, 1, 0, 1, 0, 0},
{0, 0, 1, 0, 1, 0},
{1, 0, 0, 1, 0, 1},
{1, 0, 0, 0, 1, 0}
};
for (int i = 0; i < n; i++)
for (int j = 0; j < n; j++)
graph[i][j] = temp[i][j];
for (int i = 0; i < n; i++)
colors[i] = 0;
printf("Graph Coloring using Backtracking\n");
printf("--------------------------------\n");
colorGraph(0);
if (totalSolutions == 0)
printf("No valid color assignment possible.\n");
else
printf("\nTotal Valid Combinations: %d\n", totalSolutions);
return 0;
}
```

Output:


A3-B1-01 pract 8.c
#include <stdio.h>
#define MAX 10
int n;
int graph[MAX][MAX];
int colors[MAX];
int totalSolutions = 0;
int max_colors;
int isValid(int vertex, int color) {
for (int i = 0; i < n; i++) {
if (graph[vertex][i] == 1 && colors[i] == color)
return 0;
}
return 1;
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\areeb\OneDrive\Desktop\SemIII\DAA\pract 8> & 'c:\Users\areeb\.vscode\extensions\ms-vscode.cpptools-1.28.3-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-yi5bmdkb.ww5' '--stdout=Microsoft-MIEngine-Out-naik2h5p.de5' '--stderr=Microsoft-MIEngine-Error-zpjlztgm.tuz' '--pid=Microsoft-MIEngine-Pid-v3ulaqs2.b21' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Graph Coloring using Backtracking
Solution 1: V1->C1 V2->C2 V3->C3 V4->C1 V5->C2 V6->C3
Solution 2: V1->C1 V2->C2 V3->C3 V4->C1 V5->C3 V6->C2
Solution 3: V1->C1 V2->C2 V3->C3 V4->C2 V5->C3 V6->C2
Solution 4: V1->C1 V2->C3 V3->C2 V4->C1 V5->C2 V6->C3
Solution 5: V1->C1 V2->C3 V3->C2 V4->C1 V5->C3 V6->C2
Solution 6: V1->C1 V2->C3 V3->C2 V4->C3 V5->C2 V6->C3
Solution 7: V1->C2 V2->C1 V3->C3 V4->C1 V5->C3 V6->C1
Solution 8: V1->C2 V2->C1 V3->C3 V4->C2 V5->C1 V6->C3
Solution 9: V1->C2 V2->C1 V3->C3 V4->C2 V5->C3 V6->C1
Solution 10: V1->C2 V2->C3 V3->C1 V4->C2 V5->C1 V6->C3
Solution 11: V1->C2 V2->C3 V3->C1 V4->C2 V5->C3 V6->C1
Solution 9: V1->C2 V2->C1 V3->C3 V4->C2 V5->C3 V6->C1
Solution 9: V1->C2 V2->C1 V3->C3 V4->C2 V5->C3 V6->C1
Solution 10: V1->C2 V2->C3 V3->C1 V4->C2 V5->C1 V6->C3
Solution 11: V1->C2 V2->C3 V3->C1 V4->C2 V5->C3 V6->C1
Solution 12: V1->C2 V2->C3 V3->C1 V4->C3 V5->C1 V6->C3
Solution 13: V1->C3 V2->C1 V3->C2 V4->C1 V5->C2 V6->C3
Solution 14: V1->C3 V2->C1 V3->C2 V4->C3 V5->C1 V6->C2
Solution 15: V1->C3 V2->C1 V3->C2 V4->C3 V5->C2 V6->C1
Solution 16: V1->C3 V2->C2 V3->C1 V4->C2 V5->C1 V6->C2
Solution 17: V1->C3 V2->C2 V3->C1 V4->C3 V5->C1 V6->C2
Solution 18: V1->C3 V2->C2 V3->C1 V4->C3 V5->C2 V6->C1
Total Valid Combinations: 18
PS C:\Users\areeb\OneDrive\Desktop\SemIII\DAA\pract 8>

Link:


Output Window
Compilation Results
Problem Solved Successfully
Test Cases Passed
1114 / 1114
Attempts : Correct / Total
1 / 1
Accuracy : 100%
Points Scored
4 / 4
Time Taken
0.04
Solve Next
Rat in a Maze
Black and White
Walls Coloring
Stay Ahead With:
class Solution:
def graphColoring(self, v, edges, m):
def isSafe(node, c):
def solve(node):
return solve(0)