36 Graph colouring

#include <stdio.h>

#include <limits.h>

#define MAX\_N 100

// Function to check if a color is safe for a specific node

bool isSafe(int graph[][MAX\_N], int n, int color, int v) {

for (int i = 0; i < n; i++) {

if (graph[v][i] && color == graph[i][i]) {

return false; // Color already used by an adjacent node

}

}

return true;

}

// Function to find the minimum available color for a node

int findMinAvailableColor(int graph[][MAX\_N], int n, int color[]) {

int available[MAX\_N];

// Initialize all colors as available

for (int i = 0; i < n; i++) {

available[i] = 1;

}

// Mark colors already used by adjacent nodes as unavailable

for (int i = 0; i < n; i++) {

if (graph[v][i]) {

available[color[i]] = 0;

}

}

// Find the first available color

for (int c = 1; c < n; c++) {

if (available[c]) {

return c;

}

}

// No color available (graph may not be colorable)

return -1;

}

// Recursive function to solve the graph coloring problem

bool graphColoringUtil(int graph[][MAX\_N], int n, int m, int color[], int v) {

// Base case: If all vertices are assigned a color, return true

if (v == n) {

return true;

}

// Try all possible colors for the current vertex

for (int c = 1; c <= m; c++) {

// Check if the current color is safe

if (isSafe(graph, n, c, v)) {

color[v] = c; // Assign the color

// Recursively try to color the remaining vertices

if (graphColoringUtil(graph, n, m, color, v + 1)) {

return true; // Coloring successful

}

// Backtrack if the color assignment doesn't work

color[v] = 0; // Reset the color

}

}

// No color can be assigned to the current vertex (graph may not be colorable)

return false;

}

// Function to print the solution

void printSolution(int color[], int n) {

printf("Solution:\n");

for (int i = 0; i < n; i++) {

printf(" %d ", color[i]);

}

printf("\n");

}

// Main function to solve the graph coloring problem

int main() {

/\*

Example graph (modify as needed):

1

/ \

2-----3

| |

4-----5

\*/

int graph[][MAX\_N] = {

{0, 1, 1, 1},

{1, 0, 1, 0},

{1, 1, 0, 1},

{1, 0, 1, 0}

};

int n = sizeof(graph[0]) / sizeof(graph[0][0]); // Number of vertices

int m = 3; // Number of available colors

int color[n]; // To store assigned colors

for (int i = 0; i < n; i++) {

color[i] = 0; // Initialize all color slots as unassigned

}

if (graphColoringUtil(graph, n, m, color, 0)) {

printSolution(color, n);

} else {

printf("No solution exists.\n");

}

return 0;

}