class Solution {

public:

    vector<vector<int>> floodFill(vector<vector<int>>& image, int sr, int sc, int color) {

        int m = image.size();

        int n = image[0].size();

        vector<vector<bool>> visited(m, vector<bool>(n, false));

        int initialColor = image[sr][sc];

        if(initialColor != color){

            bfs(sr, sc, visited, image, color, initialColor);

        }

        return image;

    }

    void bfs(int row, int col, vector<vector<bool>>& visited, vector<vector<int>>& image, int newColor, int initialColor) {

        int m = image.size();

        int n = image[0].size();

        queue<pair<int, int>>  q;

        q.push({row, col});

        visited[row][col] = true;

        image[row][col] = newColor;

        vector<int> dx = {0, +1, 0, -1};

        vector<int> dy = {-1, 0, +1, 0};

        while(!q.empty()){

            auto curr = q.front();

            q.pop();

            for(int i=0; i<4; i++){

                int nrow = curr.first + dx[i];

                int ncol = curr.second + dy[i];

                if(nrow >=0 && nrow<m && ncol >= 0 && ncol<n && image[nrow][ncol] == initialColor && !visited[nrow][ncol]){

                    visited[nrow][ncol] = true;

                    image[nrow][ncol] = newColor;

                    q.push({nrow, ncol});

                }

            }

        }

    }

};

<https://leetcode.com/problems/flood-fill/submissions/1324451386/>