class Solution {

public:

    int numIslands(vector<vector<char>>& grid) {

        int m = grid.size();

        int n = grid[0].size();

        int count = 0;

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

        for (int i = 0; i < m; i++) {

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

                if (!visited[i][j] && grid[i][j] == '1') {

                    count++;

                    bfs(i, j, grid, visited);

                }

            }

        }

        return count;

    }

    void bfs(int row, int col, vector<vector<char>>& grid,

             vector<vector<bool>>& visited) {

        int m = grid.size();

        int n = grid[0].size();

        queue<pair<int, int>> q;

        visited[row][col] = true;

        q.push({row, col});

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

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

        while (!q.empty()) {

            auto node = q.front();

            q.pop();

            for (int i = 0; i < 4; i++) {

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

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

                if (nrow >= 0 && nrow < m && ncol >= 0 && ncol < n &&

                    !visited[nrow][ncol] && grid[nrow][ncol] == '1') {

                    visited[nrow][ncol] = true;

                    q.push({nrow, ncol});

                }

            }

        }

    }

};