```
/*
    Time complexity: O(N * M)
    Space complexity: O(N * M)

    where N and M are the rows and columns respectively of the board
*/

void dfs(vector<vector<char>> &board, vector<vector<bool>> &visited, int x, int y, int
fromX, int fromY, char needColor, bool &foundCycle, int n, int m) {
    if (x < 0 || x >= n || y < 0 || y >= m) {
        return;
    }
    if (board[x][y] != needColor) {
        return;
    }

    if (visited[x][y]) {
        foundCycle = true;
        return;
    }

    visited[x][y] = true;

    int dx[] = {1, -1, 0, 0};
    int dy[] = {0, 0, 1, -1};

    for (int i = 0; i < 4; ++i) {
        int nextX = x + dx[i];
        int nextY = y + dy[i];
        if (nextX == fromX && nextY == fromY) {
            continue;
        }

        dfs(board, visited, nextX, nextY, x, y, needColor, foundCycle, n, m);
    }
}

bool hasCycle(vector<vector<char>> &board, int n, int m) {
    bool foundCycle = false;
    vector<vector<bool>> visited(n, vector<bool>(m, false));

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (!visited[i][j]) {
                dfs(board, visited, i, j, -1, -1, board[i][j], foundCycle, n, m);
            }
        }
    }

    return foundCycle;
}
```