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

    where N and M are the rows and columns respectively of the board
*/
import java.util.*;

public class Solution {

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

        void dfs(String[] board, int x, int y, int fromX, int fromY, char needColor, int n, int m)
        {
                if(x < 0 || x >= n || y < 0 || y >= m) return;
                if(board[x].charAt(y) != needColor) return;
                if(visited[x][y] == 1)
                {
                        findCycle = 1;
                        return;
                }
                visited[x][y] = 1;
                for(int f = 0; f < 4; f++)
                {
                        int nextX = x + dx[f];
                        int nextY = y + dy[f];
                        if(nextX == fromX && nextY == fromY) continue;
                        dfs(board, nextX, nextY, x, y, needColor, n, m);
                }
        }

        int solve(String[] board , int n, int m)
        {

                visited = new int[n][m];

                for(int i = 0; i < n ; i++)
                        Arrays.fill(visited[i],0);

                for(int i = 0; i < n; i++)
                        for(int j = 0; j < m; j++)
                                if(visited[i][j] == 0)
                                        dfs(board,i, j, -1, -1, board[i].charAt(j), n, m);

                return findCycle;
```

```
        }

}
```