CSE 4304:Data Structure Lab-08
Graph Basic

Namisa Najah Raisa 210042112

November 2023

1 Problem C:Oil Deposits

In this problem I had to find out how many oil deposits a grid contains.In the grid each plot is represented as either '*'(absence of oil) or '@'(oil pocket).Oil deposits are considered connected,and two pockets are part of the same deposit if they are adjacent horizontally, vertically,or diagonally.

```cpp
#include <iostream>
#include <vector>
using namespace std;

void dfs(vector<vector<char>>& grid, int row, int col)
{
    int m = grid.size();
    int n = grid[0].size();
    if (row < 0 || row >= m || col < 0 || col >= n ||
        grid[row][col] != '@')
    {
        return;
    }
    grid[row][col] = '*';
    dfs(grid, row - 1, col);
    dfs(grid, row + 1, col);
    dfs(grid, row, col - 1);
    dfs(grid, row, col + 1);
    dfs(grid, row - 1, col - 1);
    dfs(grid, row - 1, col + 1);
    dfs(grid, row + 1, col - 1);
    dfs(grid, row + 1, col + 1);
}
int countOilDeposits(vector<vector<char>>& grid)
{
    int m = grid.size();
    int n = grid[0].size();
    int count = 0;
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (grid[i][j] == '@')
            {
                count++;
                dfs(grid, i, j);
            }
        }
    }
    return count;
}
int main()
{
    int m, n;
    while (cin >> m >> n && m != 0)
    {
        vector<vector<char>> grid(m, vector<char>(n));
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                cin >> grid[i][j];
            }
        }
        int numDeposits = countOilDeposits(grid);
        cout << numDeposits << endl;
    }
    return 0;
}
```

I solved it using DFS.

’void dfs()’ function takes 3 parameters.It takes a grid represented as a vector of vectors of characters(’grid’), and two integers ’row’ and ’col’ referring to the current position in the grid.

’int m=grid.size()’ calculates the number of rows in the grid.It determines the maximum valid row index. ’int n=grid[0].size()’ calculates the number of columns in the grid.Determines the maximum valid column index.Assuming that all rows have the same number of columns,it’s good to access grid[0] to get the number of columns.

'if (row < 0 || row >= m || col < 0 || col >= n || grid[row][col] != '@')'→This line checks several conditions to determine whether the current position is a valid position to explore and whether it contains an oil pocket '@' or no.It does the following checks:

- 'row < 0':the position is outside the grid's upper boundary.
- 'row >= m':position is outside the grid's lower boundary.
- 'col >= n':position is outside the grid's right boundary.
- 'grid[row][col] != '@' ':checks if the current position does not contain an oil pocket. If any of these conditions are met the function returns.

'grid[row][col] = '*';'→This line marks the current position as visited by changing the character at that position from '@' to '*'.This prevents revisiting this oil pocket and ensures it's counted as part of the current deposit.

The next 8 lines represent recursive calls to the dfs() function.They explore the 8 possible adjacent positions from the current position:

1. 'dfs(grid, row - 1, col);'→explores the position above the current one.
2. 'dfs(grid, row + 1, col);'→explores the position below the current one.
3. 'dfs(grid, row, col - 1);'→explores the position left to the current one.
4. 'dfs(grid, row, col + 1);'→explores the position right to the current one.
5. 'dfs(grid, row - 1, col - 1);'→explores the position diagonally up-left.
6. 'dfs(grid, row - 1, col + 1);'→explores the position diagonally up-right.
7. 'dfs(grid, row + 1, col - 1);'→explores the position diagonally down-left.
8. 'dfs(grid, row + 1, col + 1);'→explores the position diagonally down-right.

'int countOilDeposits(vector<vector<char»& grid)'→It takes a 2D grid represented as a reference to a vector of vectors of characters as an argument.

'int count=0' This variable will keep track of the number of distinct oil deposits found in the grid.

The next FOR loop iterates through the rows of the grid.'i' is the current row.

Within the outer loop,this FOR loop iterated through the columns of the grid. 'j' is the current column.

'if (grid[i][j] == '@')'→This line checks if the current position in the grid contains oil pocket ('@').If it does the count is incremented.Then calls the dfs(grid,i,j) which will mark the current deposit and explore all connected oil pockets in a depth-first manner.After the inner loop finishes it moves to the next row and continues the process until all rows and columns in the grid have been checked.The function returns the final count variable representing the total number of distinct oil deposits found in the grid.

In the main function I took m,n inputs(row,column)

The WHILE loop continues as long as it successfully read 'm' and 'n' and 'm' is not equal to zero.Inside the loop 'vector<vector<char» grid(m, vector<char>(n));' this line declares and initializes a 2D vector called 'grid'.It creates a vector with 'm' rows and 'n' columns.

In the FOR loop that iterates through the rows there's an inner FOR loop that iterates through the columns of the grid.Inside the inner loop I took input of the grid populating it with rows and columns.

After the two nested loops end **'int numDeposits = countOilDeposits(grid);'** this line calls the countOilDeposits() function to determine the number of distinct oil deposits in the current grid.In the end it prints the number of distinct oil deposits.

________________X________________