MICROSOFT PREPARATION

DAY : 05

Number of Islands :

Problem Link : https://leetcode.com/problems/number-of-islands/

Test Cases Passed : 49 / 49

Time Used : 12.30

Difficulty Level : MEDIUM

Approach Used :

DFS() :

- Calculate the dimensions of the grid
- Mark the given node as visited
- Traverse for the adjacent elements in 4 directions :
    - Check if the indexes are valid :
        - Check if the node is unvisited and is a '1' :
            - Make a dfs call for adjacent element as dfs(adjrow,adjcol,visited,grid)

NumberOfIslands( ) :

- Calculate the dimensions of grid
- Create a visited vector of similar dimensions
- Create a counter variable to store the count of islands
- Traverse for all elements :
    - Check if the element is unvisited and is a '1' :
        - Increment the counter
        - Make a dfs call for element as dfs(row,col,visited,grid)
- Return counter

Solution :

```
void dfs(int row,int col,vector<vector<int>>&visited,vector<vector<char>>&grid)
   {
```

```
        // calculating the dimensions of the grid
        int n = grid.size();
        int m = grid[0].size();

        // mark the element as visited
        visited[row][col] = 0;

        // traverse for all its adjacent elements

        // element can be found in any of the 4 directions adjacent either
vertically or horizontally

        int delRow[] = {-1,0,1,0};
        int delCol[] = {0,1,0,-1};

        // traversing through these
        for(int i=0;i<4;i++)
        {
            // getting the dimensions of adjacent elements
            int nrow = row+delRow[i];
            int ncol = col+delCol[i];

            // checking the constraints for these indexes as they may go out of
bounds
            if(nrow<n && ncol<m && nrow>=0 && ncol>=0)
            {
                // checking if the element is unvisited and also is an island
                if(!visited[nrow][ncol] && grid[nrow][ncol]=='1')
                {
                    // make dfs call to mark it and its adjacent elements
                    dfs(nrow,ncol,visited,grid);
                }
            }
        }

    }
    int numIslands(vector<vector<char>>& grid) {
        /*

        Every connected component gives us the number of islands

        eg.

        1   1   1   0
        1   0   1   0
        1   0   0   0
        1   0   0   0 = 1 island
```

```
        mark the visited components using same or different though it will give us
the same answer every time but yeah we can encounter that which part beongs to
which island


        We will be using DFS approach to solve the particular question


        */

        // calculating the diemsnsions of the grid
        int n = grid.size();
        int m = grid[0].size();

        // Make a visited vector that will store wether the island is visited or
encountered
        vector<vector<int>> visited(n,vector<int>(m,0));

        // variable to count the number of islands
        int counter  = 0;

        // traverse for complete array and find out which of the element belong s to
a island and mark all its connected components
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<m;j++)
            {
                // if the particular element is an island and is also not visited
only then we will perform this
                if(!visited[i][j] && grid[i][j]=='1')
                {counter++;
                dfs(i,j,visited,grid);}
            }
        }
        // returning the counter
        return counter;
    }
```