MICROSOFT PREPARATION

DAY : 04

Battleships in a Board:

Problem Link : https://leetcode.com/problems/battleships-in-a-board/

Test Cases Passed : 27 / 27

Time Used : 27.19

Difficulty Level : MEDIUM

Approach Used :

dfs():

- Calculate board dimensions
- Mark the visited node with ship counter
- Traverse for all 4 adjacent directions in the ship :
    - Calculating the dimensions of the adjacent columns and rows
        - Make a dfs call to mark the neighboring cell as dfs(neighboring_row,neighboring_col,visited,board,counter)

countBattleships():

- Calculating the dimensions of the board
- Creating a visited vector
- Initializing the ship count variable with 0
- Traversing for all the components and marking the adjacent node using the dfs call as dfs(row,col,visited,board,counter)
- Returning the ship count variable

Function Code :

```
class Solution {
public:
    void dfs(int row, int col, vector<vector<int>>& visited,
vector<vector<char>>& board, int counter) {
// calculating the board dimensions
        int n = board.size();
```

```
        int m = board[0].size();
// mark the node as visited and the counter to the ship
        visited[row][col] = counter;
// traversing for the adjacent elements
        int dr[4] = {-1, 0, 1, 0};
        int dc[4] = {0, 1, 0, -1};

        for (int k = 0; k < 4; k++) {
// calculating new row and column for adjacent row and column in the four
directions
            int newRow = row + dr[k];
            int newCol = col + dc[k];
// checking for validity of adjacent row and column and checking of the //
node is visited or not and is a ‘X’
            if (newRow >= 0 && newRow < n && newCol >= 0 && newCol < m &&
                !visited[newRow][newCol] && board[newRow][newCol] == 'X') {
// making the DFS call to mark the nodes
                dfs(newRow, newCol, visited, board, counter);
            }
        }
    }

    int countBattleships(vector<vector<char>>& board) {
// calculating the board dimensions
        int n = board.size();
        int m = board[0].size();
// creating a visited vector
        vector<vector<int>> visited(n, vector<int>(m, 0));
// creating a counter variable to count the number of ships
        int counter = 0;
// traversing for all possible graph components
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
// checking if the node is not visited and board element is a ship
                if (!visited[i][j] && board[i][j] == 'X') {
// increment the ship counter and make a dfs call to mark visited
                    counter++;
                    dfs(i, j, visited, board, counter);
                }
            }
        }
// return count of ships
        return counter;
```

```
    }
};
```

Time Complexity :

O(V+E)