MICROSOFT PREPARATION

DAY : 12

Find the string in the Grid :

Problem Link : https://practice.geeksforgeeks.org/problems/find-the-string-in-grid0111/1

Test Cases Passed : 1120 / 1120

Time Used : 25.00

Difficulty Level : MEDIUM

Approach Used :

DFS():
- Calculating the dimensions of the grid
- Checking if the index is same as the word size ie. the word has been reached :
    - Return true
- Calculate the indexes of the neighboring column in the particular direction
- Check for the validity of the indexes :
    - Check if the particular direction word is same as the required word :
        - Make a dfs call to check if we can further form the word using this as dfs(adjrow,adjcol,directionrow,directioncol,index+1,word,grid) :
            - Return true because the word can be formed
- Return false because in this case the word cannot be formed in this particular direction

SearchWord():
- Calculate the dimensions of the grid
- Create a result set to contain all the possible start indexes from where the words can be formed
- Create the Delta Row and Delta Column array for all the 8 possible directions as words can only be formed in 8 directions and cannot go to other sets
- Traverse for all the elements :
    - Check if the first word of the target string is same as the letter in the grid :
        - Traverse for all the 8 directions to check if any one direction could lead to word forming :
            - Make a dfs call to check if we can form the word as dfs(row,col,directionrow,directioncol,1,word,grid) :

            - Insert the {i,j} vector into the result set because from here we can found a word in a particular direction
- Convert the result set into vector of vectors as this is the return type of the function
- Return the resulting vector

Solution :

```
bool dfs(int row, int col,int directionx,int directiony,int index, string
word, vector<vector<char>>& grid) {
    // calculating the dimensions of the grid
    int n = grid.size();
    int m = grid[0].size();
    // checking if the index is same as the word size ie. the word is
formed in a particular direction
    if (index == word.size()) {
        return true;
    }
    // calculating the indexes of the adjacent columns in a particular
direction
    int nrow = row+directionx;
    int ncol = col+directiony;
    // checking for the validity of directions and the word is same as the
upcoming word
    if(nrow<n && ncol<m && nrow>=0 && ncol>=0 &&
grid[nrow][ncol]==word[index])
    {
        // checking if the dfs can further fulfill the word
        if(dfs(nrow,ncol,directionx,directiony,index+1,word,grid))
        {
            // returning true if its possible
            return true;
        }
    }
    // return false because in this case we can never form the word in this
direction
    return false;
}

vector<vector<int>> searchWord(vector<vector<char>>& grid, string word) {
    // Calculating the dimensions of the grid
    int n = grid.size();
    int m = grid[0].size();
    // creating a result set
    set<vector<int>> result;

    // traversing for all the elements of the grid and directions
    int delRow[] = {1,0,-1,0,1,1,-1,-1};
    int delCol[] = {0,1,0,-1,-1,1,1,-1};
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            // checking if the element is the first word of the grid
            if (grid[i][j] == word[0]) {
                // traversing in particular direction and
                // calling the dfs to check if the word can be formed
starting from the given first word
                for(int k=0;k<8;k++)
                {
                     if (dfs(i, j,delRow[k],delCol[k],1, word, grid)) {
                        // inserting the index to the result set
                        result.insert({i, j});
                    }
                }
            }
        }
    }
    // converting result set to a vector
    vector<vector<int>> ans(result.begin(), result.end());
    return ans;
}
```