MICROSOFT PREPARATION

DAY : 18

Max Area of Island :

Problem Link : https://leetcode.com/problems/max-area-of-island/

Test Cases Passed : 728 / 728

Time Used : 04.10

Difficulty Level : MEDIUM

Approach Used :

DFS() :

- Calculating the dimensions of the grid
- Marking the node as visited
- Incrementing the area of component
- Traversing for adjacent elements :
    - Checking validity of indexes :
        - Checking if the component element is adjacent and unvisited :
            - Making dfs call to mark it and adjacent of it as dfs(adjrow,adjcol,visited,grid,area)

MakeConnected() :

- Calculating the dimensions of the grid
- Creating a visited vector initialized with 0
- Creating a maxarea variable to store max area initialized with 0
- Traversing through all components :
    - Checking if unvisited element ie. new component and element is 1 :
        - Calculating area of component
        - Updating max area
- Returning the max area

Solution :

```
void dfs(int row,int col,vector<vector<int>>&visited,vector<vector<int>>&grid,int
&area)
    {
        // calculating the dimensions of the grid
        int n = grid.size();
        int m = grid[0].size();
        // marking the node as visited
        visited[row][col] = 1;
        // updating the area
        area+=1;
        // traversing for the adjacent components
        int delRow[] = {-1,0,1,0};
        int delCol[] = {0,1,0,-1};
        for(int i=0;i<4;i++)
        {
            // calculating the adjacent indexes
            int nrow = row+delRow[i];
            int ncol = col+delCol[i];

            // checking for validity of the adjacent indexes
            if(nrow<n && ncol<m && nrow>=0 && ncol>=0)
            {
                // checking if the adjacent is unvisited and is an element
                if(!visited[nrow][ncol] && grid[nrow][ncol]==1)
                {
                    dfs(nrow,ncol,visited,grid,area);
                }
            }
        }
    }
    int maxAreaOfIsland(vector<vector<int>>& grid) {
        // calculating the dimensions of the grid
        int n = grid.size();
        int m = grid[0].size();
        // creating a visited vector
        vector<vector<int>> visited(n,vector<int>(m,0));
        // creating a maxarea variable and initializing it with 0
        int maxarea = 0;
        // traversing through all components to check for which component could
result in maxarea

        for(int i=0;i<n;i++)
        {
            for(int j=0;j<m;j++)
            {
```

```
                // if the component is unvisited and is a component
                if(!visited[i][j] && grid[i][j]==1)
                {
                    // calculate the area of the grid components
                    int area = 0;
                    dfs(i,j,visited,grid,area);
                    // updating the max area
                    maxarea = max(maxarea,area);
                }
            }
        }
        // returning the max area
        return maxarea;
    }
```