NAME: Ujjwal Pal

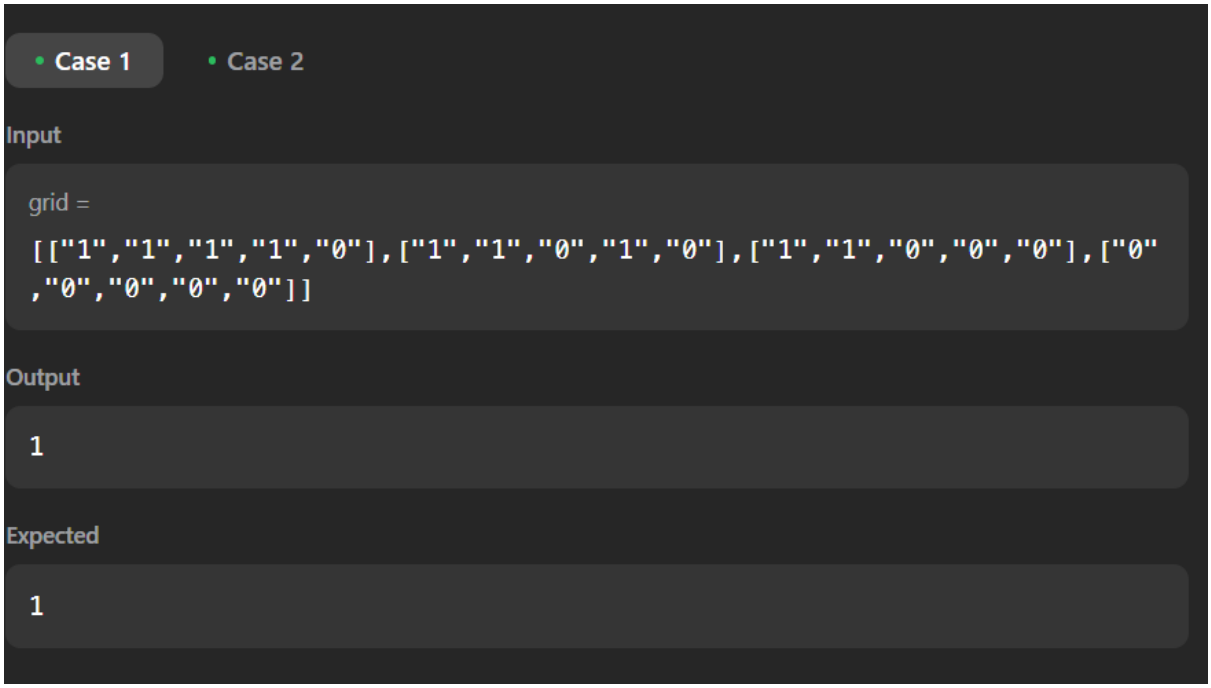
UID: 22BCS10793

Sec: 601/A

1st: 200 Number of Islands leetcode

```
class Solution {
public int numIslands(char[][] grid) {
    if (grid == null || grid.length == 0) {
        return 0;
    }

    int numIslands = 0;
    int m = grid.length;
    int n = grid[0].length;

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (grid[i][j] == '1') {
                numIslands++;
                dfs(grid, i, j);
            }
        }
    }

    return numIslands;
}

private void dfs(char[][] grid, int i, int j) {
    int m = grid.length;
    int n = grid[0].length;
```

```
        if (i < 0 || j < 0 || i >= m || j >= n || grid[i][j] == '0') {
            return;
        }


        grid[i][j] = '0';


        dfs(grid, i + 1, j);
        dfs(grid, i - 1, j);
        dfs(grid, i, j + 1);
        dfs(grid, i, j - 1);
    }
}
```



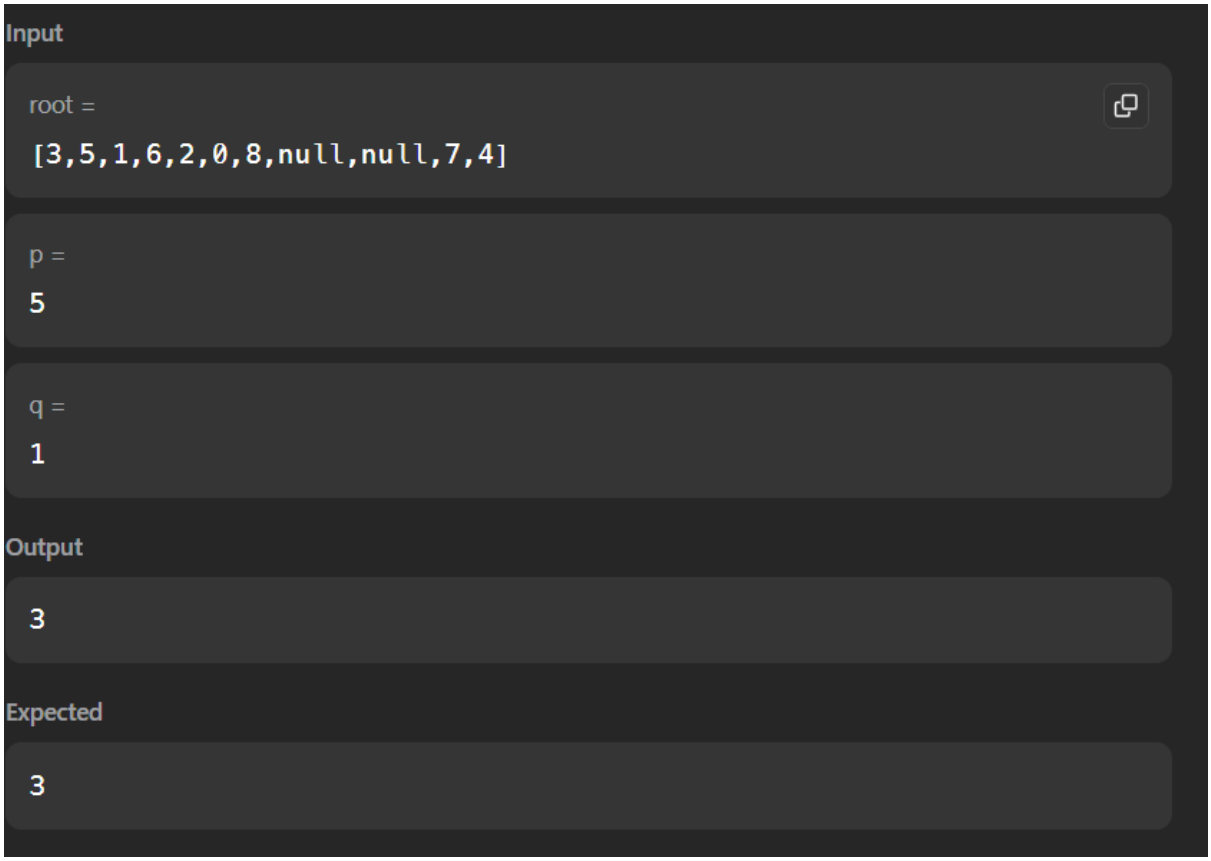
2ND:547 Number of Provinces leetcode

```
class Solution {
    boolean visited[];
    public int findCircleNum(int[][] isConnected) {
        visited=new boolean[isConnected.length];
        int cnt=0;
        for(int i=0;i<visited.length;i++)
        {
            if(!visited[i])
            {
                dfs(isConnected, i);
                cnt++;
            }
        }
        return cnt;
    }

    private void dfs(int[][] isConnected, int curr)
    {
        visited[curr]=true;
        for(int i=0;i<isConnected[curr].length;i++)
        {
            if(isConnected[curr][i]==1 && !visited[i]) dfs(isConnected, i);
        }
    }
}
```

3RD :236 Lowest Common Ancestor of a Binary Tree leetcode

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        if(root== NULL || root == p|| root ==q){
            return root;
        }

        TreeNode * left = lowestCommonAncestor(root->left, p , q);
        TreeNode * right = lowestCommonAncestor(root->right, p , q);

        if(left != NULL && right !=NULL){
            return root;
        }
```

```
        return (left != NULL) ? left : right ;
    }
};
```



4TH : 207 Course Schedule leetcode

```
class Solution {
    public boolean canFinish(int numCourses, int[][] prerequisites) {
        List<List<Integer>> graph = new ArrayList<>();
        for (int i = 0; i < numCourses; i++) {
            graph.add(new ArrayList<>());
        }



        int[] inDegree = new int[numCourses];
```

```
for (int[] prerequisite : prerequisites) {
    int course = prerequisite[0];
    int prerequisiteCourse = prerequisite[1];
    graph.get(prerequisiteCourse).add(course);
    inDegree[course]++;
}



Queue<Integer> queue = new LinkedList<>();
for (int i = 0; i < numCourses; i++) {
    if (inDegree[i] == 0) {
        queue.offer(i);
    }
}



int completedCourses = 0;



while (!queue.isEmpty()) {
    int currentCourse = queue.poll();
    completedCourses++;

    for (int dependentCourse : graph.get(currentCourse)) {
        inDegree[dependentCourse]--;

        if (inDegree[dependentCourse] == 0) {
            queue.offer(dependentCourse);
        }
    }
}
```

```
        return completedCourses == numCourses;
    }
}
```