**WEEK - 1**

1. **Given a Boolean 2D matrix, find the number of islands.**

**A group of connected 1s forms an island. For example, the below matrix contains 5 islands**

**{1, 1, 0, 0, 0},**

**{0, 1, 0, 0, 1},**

**{1, 0, 0, 1, 1},**

**{0, 0, 0, 0, 0},**

**{1, 0, 1, 0, 1}**

**A cell in the 2D matrix can be connected to 8 neighbours.**

**Use disjoint sets to implement the above scenario.**

***CODE:***

#include <stdio.h>

typedef struct DisjointUnionSets {

int rank[25];

int parent[25];

} DisjointUnionSets;

void makeSet(DisjointUnionSets \*dus, int n) {

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

dus->parent[i] = i;

dus->rank[i] = 0;

}

}

int find(DisjointUnionSets \*dus, int x) {

if (dus->parent[x] != x) {

dus->parent[x] = find(dus, dus->parent[x]);

}

return dus->parent[x];

}

void Union(DisjointUnionSets \*dus, int x, int y) {

int xRoot = find(dus, x);

int yRoot = find(dus, y);

if (xRoot == yRoot) {

return;

}

if (dus->rank[xRoot] < dus->rank[yRoot]) {

dus->parent[xRoot] = yRoot;

} else if (dus->rank[yRoot] < dus->rank[xRoot]) {

dus->parent[yRoot] = xRoot;

} else {

dus->parent[yRoot] = xRoot;

dus->rank[xRoot]++;

}

}

int countIslands(int a[][5], int n, int m) {

DisjointUnionSets dus;

makeSet(&dus, n \* m);

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

for (int k = 0; k < m; k++) {

if (a[j][k] == 0) {

continue;

}

if (j + 1 < n && a[j + 1][k] == 1) {

Union(&dus, j \* (m) + k, (j + 1) \* (m) + k);

}

if (j - 1 >= 0 && a[j - 1][k] == 1) {

Union(&dus, j \* (m) + k, (j - 1) \* (m) + k);

}

if (k + 1 < m && a[j][k + 1] == 1) {

Union(&dus, j \* (m) + k, (j) \* (m) + k + 1);

}

if (k - 1 >= 0 && a[j][k - 1] == 1) {

Union(&dus, j \* (m) + k, (j) \* (m) + k - 1);

}

if (j + 1 < n && k + 1 < m && a[j + 1][k + 1] == 1) {

Union(&dus, j \* (m) + k, (j + 1) \* (m) + k + 1);

}

if (j + 1 < n && k - 1 >= 0 && a[j + 1][k - 1] == 1) {

Union(&dus, j \* m + k, (j + 1) \* (m) + k - 1);

}

if (j - 1 >= 0 && k + 1 < m && a[j - 1][k + 1] == 1) {

Union(&dus, j \* m + k, (j - 1) \* m + k + 1);

}

if (j - 1 >= 0 && k - 1 >= 0 && a[j - 1][k - 1] == 1) {

Union(&dus, j \* m + k, (j - 1) \* m + k - 1);

}

}

}

int c[25] = {0};

int numberOfIslands = 0;

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

for (int k = 0; k < m; k++) {

if (a[j][k] == 1) {

int x = find(&dus, j \* m + k);

if (c[x] == 0) {

numberOfIslands++;

c[x]++;

} else {

c[x]++;

}

}

}

}

return numberOfIslands;

}

int main(void) {

int a[5][5] = {

{1, 1, 0, 0, 0},

{0, 1, 0, 0, 1},

{1, 0, 0, 1, 1},

{0, 0, 0, 0, 0},

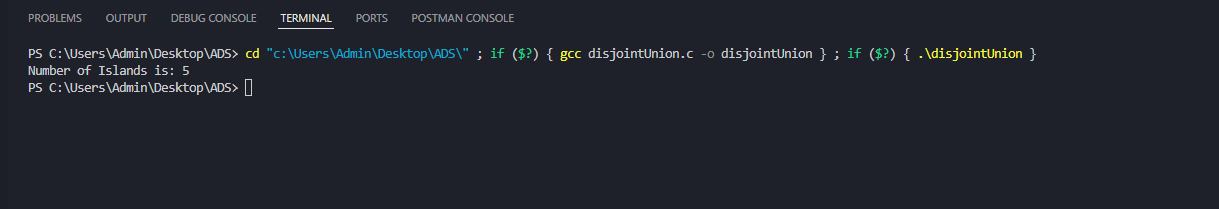
{1, 0, 1, 0, 1}

};

printf("Number of Islands is: %d\n", countIslands(a, 5, 5));

return 0;

}

**OUTPUT:**