Name: Anusree Manoj K
USN: 1BM18CS017
Date: 7-10-2020

LAB 4 Program 3 - No. of Islands using disjoint set.

Algorithm:

1. Convert m*n (m rows n columns) matrix (mat[][]) to 1D array (parent[]) of length m*n.
   For each mat[i][j], match (i,j) to (n*i+j)
   so index (n*i+j) represents of mat[i][j],
   parent[n*i+j] represents which subset the mat[i][j] belongs to
2. Count all islands (int count)
3. Loop through the matrix (2D) (mat[][]),
   if find an island X (points to root parent element S),
   check the adjacent neighbours. If any adjacent island present, it should be in the same subset of
   ~~X If the root parent~~
   If there is an adjacent island Y and is not in the same subset of X, i.e., the root parent element of Y is not S, then merge Y to subset S by setting Y as the parent element of S and count--; (Union operation)
4. While one island is merged to a subset, the number of island will be decremented by 1.
   After we visit all the connected islands, we get the no. of islands.

Anusree

Anuree Manoj K
IBM18CS017

```
class UnionFind {
    int [] parent;
    int count;

public UnionFind(int n) {
        parent[n];
        for (int i=0; i<n; i++){
            parent[i]=i;
        }
        count=0;
}
public int find(int x) {
    if (parent[x]==x) {
        return x;
    }
    return parent[x] = find(parent[x]);
}
public void connect (int x, int y){
    int rootx = find(x)
    int rooty = find(y)
    if (rootx != rooty){
        parent[rootx] = rooty;
        count--;
    }
}
```


Anuree

Sreesree Manoj K
1BM18CS017

```
public void setCount(int n){
    count=n;
}
public int count(){
    return count;
}
};
```

~~class~~
~~public class Solution~~
~~int numIslands(char[][] mat)~~
~~if (mat==null)~~
~~int m= mat.length;~~

```
int numIslands(vector<vector<int>> mat)
{
    int count=0;
    int m= mat.size();
    int n= mat[0].size();
    for(int i=0; i<m; i++){
        for(int j=0; j<n; j++){
            if(mat[i][j])
                count++;
    }}
    UnionFind uf= new UnionFind(m*n);
    uf.setCount(count);
    for(int i=0; i<m; i++){
        for(int j=0; j<n; j++){
            if(mat[i][j]){
```


sreesree

```
if (i>0 && mat[i-1][j]) {
    uf.connect(n*i+j, n*(i-1)+j);
}
if(i<m-1 && mat[i+1][j]) {
    uf.connect(n*i+j, n*(i+1)+j);
}
if(j>0 && mat[i][j-1]) {
    uf.connect(n*i+j, n*i+j-1);
}
if(j<n-1 && mat[i][j+1]) {
    uf.connect(n*i+j, n*i+j+1);
}
if(i>0 && j>0 && mat[i-1][j-1]) {
    uf.connect(n*i+j, n*(i-1)+j-1);
}
if(i<m-1 && j<n-1 && mat[i+1][j+1]) {
    uf.connect(n*i+j, n*(i+1)+j+1);
}
if(i>0 && j<n-1 && mat[i-1][j+1]) {
    uf.connect(n*i+j, n*(i-1)+j+1);
}
if(i<m-1 && j>0 && mat[i+1][j-1]) {
    uf.connect(n*i+j, n*(i+1)+j-1);
}
}
}
}
return uf.count();
}
}
```


Anusree