Name : Harshit Hiremath

USN : 1BM18CS036

Numbers of islands using Disjoint sets

1. Get 1D array (parent [ ]) from matrix [ ][ ] of length m*n.

For each mat [i] [j] match(i, j) to (n*i +j).

Index (n*i +j) represents mat[i] [j], parent [n*i +j] represents which subset the mat[i] [j) belongs

2. Count the islands

3. Loop through matrix mat[ ][ ).

if found island x (points to root parent s),

check the adjacent neighbours.

If any present, they should be in same subset.

If any present, and not in same subset, 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, the number of island ~~whi~~ will be decremented by 1. After we unite them all, we get the number of islands.

Harshit Hiremath

Psuc-

```
class _union find {
      int parent [];
      int count;
   public: unionFind (int n) {
            parent [n];
            for (i=0; i<n; i++){
                  parent (i) =i
            }
            Count=0
      }

   int  ~~int~~ find (int x) {
         if (parent [x]==x) {
               return x
         }
         return parent [x]= find (parent [])
   }
   void connect (int x, int y) {
         int rootx= find(x)
         int root y = find (y)
         if ( rootx != rooty) {
               parent [rootx] = rooty;
               count --;
         }
   }
```

Harshit Hiremath

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

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.setc~~
    uf.setCount(count);
    for (int i=0; i<m; i++) {
        for (int j=0; j<n; j++){
            if (mat[i][j]){
                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.connect();
}
```