7/10/20

LAB - 4

Abhiram G
1BM18CS127

Procedure:

-> Initialise no. of islands = 0
-> Traverse 2D matrix
-> If value 1, check its neighbours, if neighbour 1 take union
-> Define array of size row x column
-> Traverse matrix
-> if value = 1, find its set
-> If freq of set 0, increment result to 1.

```
int noofIslands (vector <vector< int >>a) {
  int n -> a.size();
  int m -> a[0].size();
Disjoint Union Sets *dus = new Dicjoint Union Sets (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
      dus -> Union (j * (m) + k, (j+1) * (m) +k);
   if j-1 >=0 && a[j-1][k] == 1
      dus -> Union (j * (m) + k , (j-1) * (m) +k);
   if k+1 < m && a[j][k+1] == 1
      dus -> Union (j * (m) + k, (j) * m + k + 1);
   if k-1 >=0 && a[j][k-1] == 1
      dus -> Union (j * (m) + k, (j) * (m) + k-1);
   if j+1 < n && k+1 < m &&
         a[j+1][k+1] == 1)
   dus -> Union (j * (m) + k, (j+1) * (m) + k+1);
```

```
        if (j+1 < n && k-1 >= 0 &&
            a[j+1][k-1] == 1)
          dus -> Union (j*m+k, (j+1)*(m)+k-1);
        if (j-1 >= 0 && k+1 < m &&
            a[j-1][k+1] == 1)
          dus -> Union (j*m+k, (j-1)*m+k+1);
        if (j-1 >= 0 && k-1 >= 0 &&
            a[j-1][k-1] == 1)
          dus -> Union (j*m+k, (j-1)*m+k-1);
      int *c = new int[n*m];
      int number Of Islands = 0;
      for (int j -> 0; j < n; j++)
      {
        for (int k -> 0; k < m; k++)
        {
          if a[j][k] == 1
          {
            int x = dus -> find (j*m+k);
          if (c[x] == 0)
            number of islands++;
            c[x]++;
          else
            c[x]++;
      return number of Islands;
```