```
class   DisjointUnionSets
{
    vector <int>   rank, parent;
    int   n;

    public:
    DisjointUnionSets (int n)
    {
        rank.resize(n);
        parent.resize (n);
        this -> n  = n;
        makeSet ();
    }
   void   makeSet ()
   {
       for (int i=0 ; i<n ; i++ )
                parent [i] = i ;
   }

 int  find(int x )
 {
      if  (parent [x] != x )
           return   find (parent [x])
      return x ;

 }
void   Union (int x, y)
{
      int    xRoot = find(x);
      int    yRoot = find(y);

      if (xRoot == yRoot)
            return;

      if (rank[xRoot] < rank[yRoot])
            parent [xRoot] = yRoot;
   else if (rank[yRoot] < rank[xRoot])
            parent [yRoot] = xRoot;
      else {
            parent[yRoot] = xRoot;
            rank[xRoot] = rank[xRoot] + 1;
      }
   }
};
```

```
int main()
{
    vector<vector<int>> a = {{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}};
    cout<< "Number of islands is : " <<
          countIslands(a) << endl;
}


nt  countIslands(vector<vector<int>> a)

   int  n = a.size();
   int  m = a[0].size();

DisjointUnionSets  *dus = new DisjointUnionSets(n*

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, a (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, (1+j)* 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 >= 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))

 }
}
```

```
int   c =  new  int [n*m] ;
int   numberOfIslands = 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)
           {
                   numberOf Islands ++ ;
                   c [x]++ ;
           }
           else
               c [x] ++ ;
      }
   }
}
return  numberOf Islands ;

}
```