Given a boolean 2D-Matrix, find the no of islands. A group of connected 1's forms an island.
For eg, 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.

```
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, int 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 countIslands(vector<vector<int>>a)
{
    int n = a.size();
    int m = a[0].size();

    DisjointUnionSets *ds = new DisjointUnionSets(
```

```
int *c = new int [n*m];
    int numberOfIslands = 0;

for (int j = 0; j<n ; j++)
  {
    for (k=0 ; k<m ; k++)
      if(a[j][k]==1)
        {
          int x = dus⇒find (j*m+k);
          if (c[x]==0)
          {
            numberOfIslands++;
            c[x]++;
          }
          else
            c[x]++;
        }
  }
}
return numberOfIslands;
}
```

```
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);
    }
}
```

Date
Page

Approach:

1) Initialize result (count of islands) as 0
2) Traverse each index of the 2D matrix
3) If the value at index is 1, check all its 8 neighbours. If a neighbour is also equal to 1, take the union of index & its neighbour
4) Now define an array of size row * column to store frequencies of all sets.
5) Now traverse the matrix again
6) If the value at the index is 1, find its set.
7) If the frequency of the set in the above array is 0, increment the result be 1.