ISLANDS IN A TWO DIMENSIONAL ARRAY

Each element in the array is connected to eight other elements (toward top, down, left and right). For example, the zero below is connected to all the ones

```
1 1 1

1 0 1

1 1 1
```

Given a two-dimensional array of 0's and 1's. Find the number of islands where an island is a group of connected 1's. For example: The below array has 4 islands (shown in different colours)

```
1 1 0 0 0
0 1 0 0 1
1 0 0 1 1
0 0 0 0 0
1 1 1 0 1
```

Write code which will accept this 2-dim array and return the number of islands.

Solution:

The problem is similar to the problem of connected components in graph.

The solution is simple. Look for 1 in the array which is not connected to any other graph, when you find 1 search for the all the entries in this island by recursively searching the eight connected nodes, and mark them connected.

In the code let us set the value to -1 to indicate that the node is already connected to an island.

Let's first write the recursive function to mark the elements as connected. It will take a position (i,j) and mark it -1 and then look for nearby cells and mark all the nodes that are 1 as -1. And also call the function recursively.

```
1    void markIsland(int arr[][5], int i, int j, int m, int n)
2    {
3        arr[i][j] = -1;
4
5        if(i-1 >= 0)
6        {
7            // (i-1, j-1)
8            if(j-1 >= 0 && arr[i-1][j-1] == 1)
9                markIsland(arr, i-1, j-1, m, n);
10
```

```
11          // (i-1, j)
12          if(arr[i-1][j] == 1)
13              markIsland(arr, i-1, j, m, n);
14
15          // (i-1, j+1)
16          if(j+1 < n && arr[i-1][j+1] == 1)
17              markIsland(arr, i-1, j+1, m, n);
18      }
19
20      if(i+1 < m)
21      {
22          // (i+1, j-1)
23          if(j-1 >= 0 && arr[i+1][j-1] == 1)
24              markIsland(arr, i+1, j-1, m, n);
25
26          // (i+1, j)
27          if(arr[i+1][j] == 1)
28              markIsland(arr, i+1, j, m, n);
29
30          // (i+1, j+1)
31          if(j+1 < n && arr[i+1][j+1] == 1)
32              markIsland(arr, i+1, j+1, m, n);
33      }
```

```
34
35        // (i, j-1)
36        if(j-1 >= 0 && arr[i][j-1] == 1)
37            markIsland(arr, i, j-1, m, n);
38
39        // (i, j+1)
40        if(j+1 < n && arr[i][j+1] == 1)
41            markIsland(arr, i, j+1, m, n);
42    }
```

The main function that count the islands will just call this function to mark all the elements in the islands whenever a 1 is encountered.

```
1     int countIslands(int arr[][5], int m, int n)
2     {
3         int count = 0;
4
5         for(int i=0; i<m; i++)
6             for(int j=0; j<n; j++)
7                 if(arr[i][j] == 1)
8                 {
9                     count++;
10                    markIsland(arr, i, j, m, n);
11                }
```

```
12

13          return count;

14      }
```

The only problem with above code is that it changes all the 1's to -1. But that can easily be corrected by adding one more loop to the function

```
1       for(int i=0; i<m; i++)

2               for(int j=0; j<n; j++)

3                       if(arr[i][j] == -1)

4                               arr[i][j] = 1;
```

Time complexity: O(M*N).

Extra space will be used because of recursion.

Source: http://www.ritambhara.in/islands-in-a-two-dimensional-array/