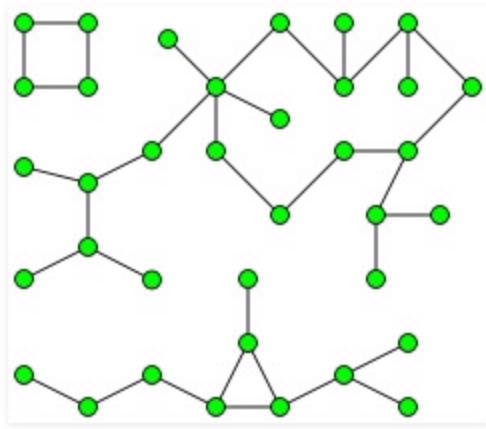
Find the number of islands | Set 1 (Using DFS)

Given a boolean 2D matrix, find the number of islands. A group of connected 1s forms an island. For example, the below matrix contains 5 islands

```
Input : mat[][] = {{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}
Output : 5
```

This is an variation of the standard problem: "Counting number of connected components in a undirected graph".

Before we go to the problem, let us understand what is a connected component. A connected component of an undirected graph is a subgraph in which every two vertices are connected to each other by a path(s), and which is connected to no other vertices outside the subgraph.
For example, the graph shown below has three connected components.



We strongly recommend that you click here and practice it, before moving on to the solution.

A graph where all vertices are connected with each other, has exactly one connected component, consisting of the whole graph. Such graph with only one connected component is called as Strongly Connected Graph.

The problem can be easily solved by applying DFS() on each component. In each DFS() call, a component or a sub-graph is visited. We will call DFS on the next un-visited component. The number of calls to DFS() gives the number of connected components. BFS can also be used.

What is an island?

A group of connected 1s forms an island. For example, 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 2D matrix can be connected to 8 neighbors. So, unlike standard DFS(), where we recursively call for all adjacent vertices, here we can recursive call for 8 neighbors only. We keep track of the visited 1s so that they are not visited again.

C/C++ Java Python

```
// Program to count islands in boolean 2D matrix
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

#define ROW 5
#define COL 5

// A function to check if a given cell (row, col) can be included in DFS
int isSafe(int M[][COL], int row, int col, bool visited[][COL])
{
    // row number is in range, column number is in range and value is 1
    // and not yet visited
    return (row >= 0) && (row < ROW) &&
           (col >= 0) && (col < COL) &&
           (M[row][col] && !visited[row][col]);
}

// A utility function to do DFS for a 2D boolean matrix. It only considers
// the 8 neighbours as adjacent vertices
void DFS(int M[][COL], int row, int col, bool visited[][COL])
{
    // These arrays are used to get row and column numbers of 8 neighbours
    // of a given cell
    static int rowNbr[] = {-1, -1, -1,  0, 0,  1, 1, 1};
    static int colNbr[] = {-1,  0,  1, -1, 1, -1, 0, 1};

    // Mark this cell as visited
    visited[row][col] = true;

    // Recur for all connected neighbours
    for (int k = 0; k < 8; ++k)
        if (isSafe(M, row + rowNbr[k], col + colNbr[k], visited) )
            DFS(M, row + rowNbr[k], col + colNbr[k], visited);
}

// The main function that returns count of islands in a given boolean
// 2D matrix
int countIslands(int M[][COL])
{
    // Make a bool array to mark visited cells.
    // Initially all cells are unvisited
    bool visited[ROW][COL];
    memset(visited, 0, sizeof(visited));

    // Initialize count as 0 and travese through the all cells of
    // given matrix
    int count = 0;
    for (int i = 0; i < ROW; ++i)
        for (int j = 0; j < COL; ++j)
            if (M[i][j] && !visited[i][j]) // If a cell with value 1 is not
            {                              // visited yet, then new island found
                DFS(M, i, j, visited);     // Visit all cells in this island.
                ++count;                   // and increment island count
            }

    return count;
}

// Driver program to test above function
int main()
{
    int M[][COL]= {  {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}
    };

    printf("Number of islands is: %d\n", countIslands(M));

    return 0;
}
```

Run on IDE

Output:

```
Number of islands is: 5
```

Time complexity: O(ROW x COL)Find the number of Islands | Set 2 (Using Disjoint Set)

Asked in: Microsoft, Amazon, Citrix, Informatica, Ola, One97, Opera, Paytm, Snapdeal, Streamoid Technologi, Visa

Reference:
http://en.wikipedia.org/wiki/Connected_component_%28graph_theory%29