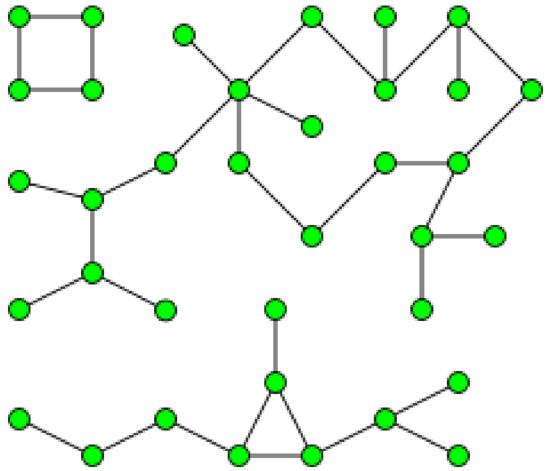
Find the number of islands

Given a boolean 2D matrix, find the number of islands.

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.



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.

GeeksforGeeks
53,520 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

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.

```
// 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 D
int isSafe(int M[][COL], int row, int col, bool visited[][COL])
{
    return (row >= 0) && (row < ROW) &&     // row number is in range
           (col >= 0) && (col < COL) &&     // column number is in ran
           (M[row][col] && !visited[row][col]); // value is 1 and not
}

// A utility function to do DFS for a 2D boolean matrix. It only consi
// the 8 neighbors 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 neighb
    // 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};
```

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
    // 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 i
            {                              // visited yet, then new is
                DFS(M, i, j, visited);     // Visit all cells in this
                ++count;                   // and increment island cou
            }

    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;
}
```

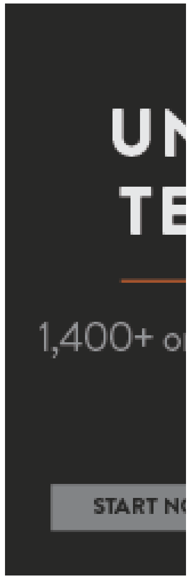
Output:

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

Time complexity: O(ROW x COL)

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

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum

705 Subscribe

Recent Comments

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 9 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 12 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 37 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 39 minutes ago

newCoder3006 If the array contains negative numbers also. We...
Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...
Find subarray with given sum · 1 hour ago

AdChoices

► C++ Vector
► Matrix in Java
► C++ Code

- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

8 Tweet 0 0

Writing code in comment? Please use **ideone.com** and share the link here.