**EXP 5:**

BFS and DFS

# Divyansh Sinha

# RA1811030010045

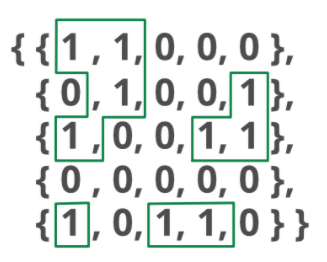
**Problem taken:**

Counting the number of islands

**Description:**

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

Example:-



What is a connected Graph?

 A [connected component](https://www.geeksforgeeks.org/connected-components-in-an-undirected-graph/) of an undirected graph is a subgraph in which every two vertices are connected to each other by a path/paths, and which is connected to no other vertices outside the subgraph.  
For example, the graph shown below has three connected components



**Solution:**

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.

**Code :**

class Graph:

def \_\_init\_\_(self, row, col, g):

self.ROW = row

self.COL = col

self.graph = g

# A function to check if a given cell

# (row, col) can be included in DFS

def isSafe(self, i, j, visited):

# row number is in range, column number

# is in range and value is 1

# and not yet visited

return (i >= 0 and i < self.ROW and

j >= 0 and j < self.COL and

not visited[i][j] and self.graph[i][j])

# A utility function to do DFS for a 2D

# boolean matrix. It only considers

# the 8 neighbours as adjacent vertices

def DFS(self, i, j, visited):

# These arrays are used to get row and

# column numbers of 8 neighbours

# of a given cell

rowNbr = [-1, -1, -1, 0, 0, 1, 1, 1];

colNbr = [-1, 0, 1, -1, 1, -1, 0, 1];

# Mark this cell as visited

visited[i][j] = True

# Recur for all connected neighbours

for k in range(8):

if self.isSafe(i + rowNbr[k], j + colNbr[k], visited):

self.DFS(i + rowNbr[k], j + colNbr[k], visited)

# The main function that returns

# count of islands in a given boolean

# 2D matrix

def countIslands(self):

# Make a bool array to mark visited cells.

# Initially all cells are unvisited

visited = [[False for j in range(self.COL)]for i in range(self.ROW)]

# Initialize count as 0 and travese

# through the all cells of

# given matrix

count = 0

for i in range(self.ROW):

for j in range(self.COL):

# If a cell with value 1 is not visited yet,

# then new island found

if visited[i][j] == False and self.graph[i][j] == 1:

# Visit all cells in this island

# and increment island count

self.DFS(i, j, visited)

count += 1

return count

graph = [[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]]

row = len(graph)

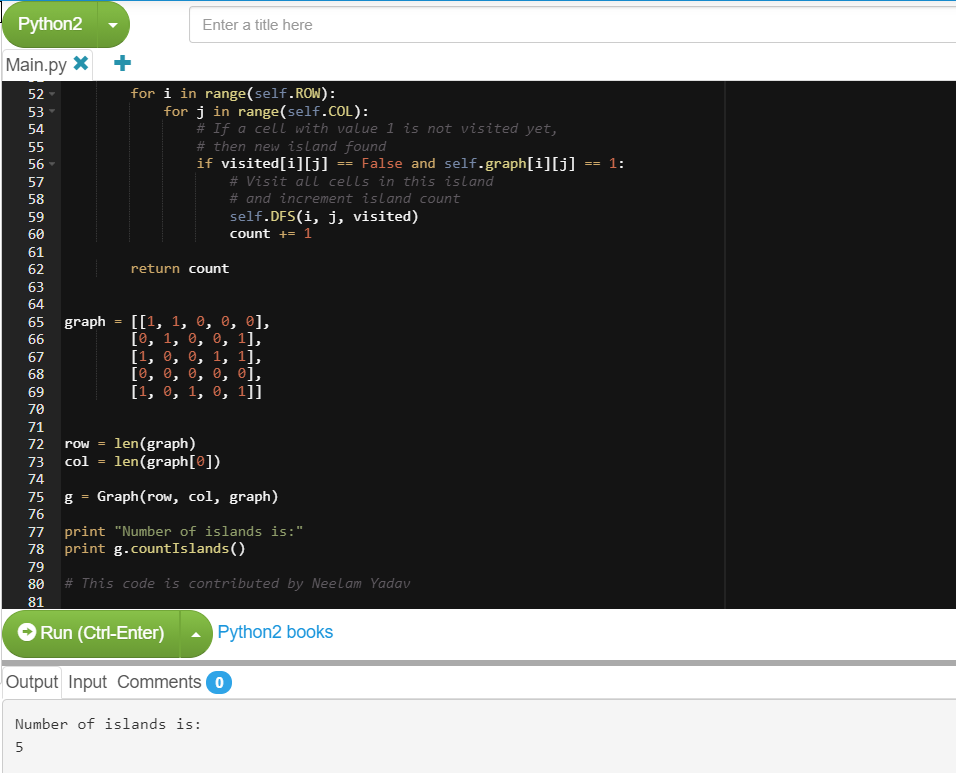
col = len(graph[0])

g = Graph(row, col, graph)

print "Number of islands is:"

print g.countIslands()

**Output:**



**Time Complexity**:

O(ROW x COL)