[**Number of Islands**](https://leetcode.com/problems/number-of-islands/)

**package** leetcodeMedium;

**public** **class** NumberOfIslands {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**char**[][] grid = {{'1','1','0','0','0'},{'1','1','0','0','0'},{'0','0','1','0','0'},{'0','0','0','1','1'}};

System.***out***.println(*numIslands*(grid));

}

**public** **static** **int** numIslands(**char**[][] grid) {

**if**(grid == **null** || grid.length == 0 || grid[0].length == 0) {

**return** 0;

}

**int** count = 0;

**for**(**int** i = 0 ; i < grid.length ; i++) {

**for**(**int** j = 0 ; j < grid[0].length ; j++) {

**if**(grid[i][j] != '0') {

count += *dfs*(grid , i , j);

}

}

}

**return** count;

}

**public** **static** **int** dfs(**char**[][] grid , **int** i , **int** j) {

**if**(i < 0 || j < 0 || i >= grid.length || j >= grid[0].length || grid[i][j] == '0') {

**return** 0;

}

**int** result = 0;

grid[i][j] = '0';

result++;

*dfs*(grid , i + 1 , j);

*dfs*(grid , i - 1 , j);

*dfs*(grid , i , j + 1);

*dfs*(grid , i , j - 1);

**return** result;

}

}

Time Complexity : O(n ^ 2) , n is length of grid

Space Complexity : O(1), constant space