

Number of Islands & Leetcode (200)

Question: $m \times n$ 2D binary grid

1 \rightarrow land 0 \rightarrow water

An island is surrounded by water formed by connecting adjacent lands horizontally / vertically & not diagonally.
We can assume all 4 edges of grid are surrounded by water

Approach:

- Start from the leftmost topmost point of grid
- if you encounter a land (1) keep looking around in all possible direction for adjacent lands and continue to do this until an island is formed.
- Once an island is found increase the count of no. of islands.
- Traversal is done? \rightarrow for every row check every column.
So if its already visited in the previous call of dfs (make sure it was changed from 1 to 0).
- So when you encounter '1' it was definitely not part of the previous island. Start the dfs again.
- At last return the count (no. of islands).

Pseudocode:

Main fn

for every row
check every column
if land (1)
call dfs fn.
increase count

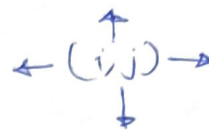
note count will be increased when for a (i,j)
dfs is done on all possible ways completely

Dfs fn (private)

if (i,j) ∈ grid & grid[i][j] = 1
(land)

change it to '0'
grid[i][j] = 0

then do dfs on possible
adjacent cells.



Code:

Main fn

```
public int numIslands(char[][] grid) {
    int count = 0;
    for (int i = 0; i < grid.length; i++)
        for (int j = 0; j < grid[0].length; j++) {
            if (grid[i][j] == '1') {
                dfsFill(grid, i, j);
                count++;
            }
        }
    return count;
}
```

it is a character

Dfs fn

```
private void dfsFill(char[][] grid, int i, int j) {
    if (i < 0 || j < 0 || i > grid.length || j > grid[0].length || grid[i][j] == '0')
        return;
    grid[i][j] = '0';
    dfsFill(grid, i+1, j);
    dfsFill(grid, i-1, j);
    dfsFill(grid, i, j+1);
    dfsFill(grid, i, j-1);
}
```