

200. Number of Islands

Given an $m \times n$ 2D binary grid `grid` which represents a map of `'1'`'s (land) and `'0'`'s (water), return *the number of islands*.

An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:

```
Input: grid = [
  ["1","1","1","1","0"],
  ["1","1","0","1","0"],
  ["1","1","0","0","0"],
  ["0","0","0","0","0"]
]
Output: 1
```

Example 2:

```
Input: grid = [
  ["1","1","0","0","0"],
  ["1","1","0","0","0"],
  ["0","0","1","0","0"],
  ["0","0","0","1","1"]
]
Output: 3
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 300`
- `grid[i][j]` is `'0'` or `'1'`.

```
class Solution:
    def numIslands(self, grid: List[List[str]]) -> int:
        res = 0
        visited = [[False for j in range(len(grid[0]))] for i in
range(len(grid))]
```

```

R = len(grid)
C = len(grid[0])
for i in range(len(grid)):
    for j in range(len(grid[0])):
        if visited[i][j]==False and grid[i][j]=='1':
            self.dfs(grid,visited,i,j)
            res = res+1
return res

def dfs(self,grid,visited,r,c):
    if r<0 or c<0 or r>=len(grid) or c>=len(grid[0]) or visited[r]
[c]==True or grid[r][c]=='0':
        return
    visited[r][c] = True
    self.dfs(grid,visited,r-1,c)
    self.dfs(grid,visited,r,c+1)
    self.dfs(grid,visited,r+1,c)
    self.dfs(grid,visited,r,c-1)

```

It is just the application of Connected components question.