**200. Number of Islands**
Medium

Given an m x 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 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
```

trick:
use a set() to keep track of visited islands
then use BFS or DFS

```python
class Solution:
    def numIslands(self, grid: List[List[str]]) -> int:
        if not grid:
            return 0

        rows, cols = len(grid), len(grid[0])

        visit = set()
        islands = 0

        def bfs(r, c):
            q = collections.deque()
            visit.add((r, c))
            q.append((r, c))
            while q:
                rw, cl = q.popleft()
                directions = [[1, 0], [-1, 0], [0, 1], [0, -1]]
                for dr, dc in directions:
                    r, c = rw + dr, cl + dc
                    if (r in range(rows) and
                        c in range(cols) and
                        grid[r][c] == "1" and
                        (r, c) not in visit):
                        q.append((r, c))
                        visit.add((r,c))

    for row in range(rows):
        for col in range(cols):
            if (grid[row][col] == "1" and
                (row, col) not in visit):
                bfs(row, col)
                islands += 1
    return islands
```

visit each row/col

update the `visit` set, and follow the path of land
and update the `islands` variable

```python
class Solution:
    def numIslands(self, grid: List[List[str]]) -> int:
            if not grid:
                return 0
        rows, cols = len(grid), len(grid[0])

        visit = set()
        islands = 0

        def dfs(r, c):
            if grid[r][c] == "0" or (r,c) in visit:
                return
            visit.add((r, c))
            if 0 <= r + 1 < rows and 0 <= c < cols:
                dfs(r + 1, c)
            if 0 <= r - 1 < rows and 0 <= c < cols:
                dfs(r - 1, c)
            if 0 <= r < rows and 0 <= c + 1 < cols:
                dfs(r, c + 1)
            if 0 <= r < rows and 0 <= c - 1 < cols:
                dfs(r, c - 1)

    for row in range(rows):
        for col in range(cols):
            if (grid[row][col] == "1" and (row, col) not in visit):
                dfs(row, col)
                islands += 1

    return islands
```

```
grid = [
  ["1","1","0","0","0"],
  ["1","1","0","0","0"],
  ["0","0","1","0","0"],
  ["0","0","0","1","1"]
]
```