**Coding Challenge 1:**

**Surrounded Regions**

Description: Replace 'O' regions surrounded by 'X' in a 2D grid with 'X'.

**Problem Statement:**

You are given an m x n matrix board containing letters 'X' and 'O'. Capture all regions surrounded by 'X'. A region is considered surrounded if all 'O's are completely enclosed by 'X' and none are on the border. To capture a region, replace all 'O's in the surrounded region with 'X's in-place.

**Input Format:** - board: List of m rows, each with n characters 'X' or 'O'

**Constraints:**

- 1 <= m, n <= 200  
- board[i][j] is either 'X' or 'O'

**Output Format:** Return the modified board with surrounded regions captured.

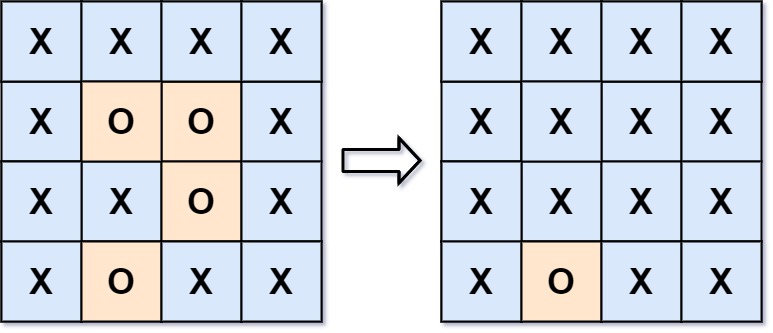
**Tags:** Matrix, DFS, BFS, Union Find, Flood Fill

**Test case 1:**

**Input:** board = [["X","X","X","X"],["X","O","O","X"],["X","X","O","X"],["X","O","X","X"]]

**Output:** [["X","X","X","X"],["X","X","X","X"],["X","X","X","X"],["X","O","X","X"]]

**Explanation:**



In the above diagram, the bottom region is not captured because it is on the edge of the board and cannot be surrounded.

**Test case 2:**

**Input:** board = [["X"]]

**Output:** [["X"]]

Test Case 3:  
Input: [["O","O"],["O","O"]]  
Output: [["O","O"],["O","O"]]

Test Case 4:  
Input: [["X","O","X"],["O","O","O"],["X","O","X"]]  
Output: [["X","O","X"],["O","O","O"],["X","O","X"]]

Test Case 5:  
Input: [["X","X","X"],["X","O","X"],["X","X","X"]]  
Output: [["X","X","X"],["X","X","X"],["X","X","X"]]

**Coding Challenge 2:**

**Course Schedule**

Description: Check if all courses can be finished given prerequisites.

**Problem Statement:**

You are given a number of courses labeled from 0 to numCourses - 1, and a list of prerequisite pairs. Each pair [a, b] indicates you must take course b before course a. Return true if it's possible to finish all courses; otherwise, return false.

**Input Format:**

- numCourses: Integer  
- prerequisites: List of pairs [a, b]

**Constraints:**

- 1 <= numCourses <= 2000  
- 0 <= prerequisites.length <= 5000  
- prerequisites[i].length == 2  
- 0 <= a, b < numCourses  
- All prerequisite pairs are unique

**Output Format:**

Return true or false.

**Tags:**

Graph, Topological Sort, DFS, Cycle Detection

**Test Cases:**

Test Case 1:  
Input: numCourses = 2, prerequisites = [[1, 0]]  
Output: true

Test Case 2:  
Input: numCourses = 2, prerequisites = [[1, 0], [0, 1]]  
Output: false

Test Case 3:  
Input: numCourses = 3, prerequisites = [[1, 0], [2, 1]]  
Output: true

Test Case 4:  
Input: numCourses = 1, prerequisites = []  
Output: true

Test Case 5:  
Input: numCourses = 4, prerequisites = [[1,0],[2,1],[3,2],[1,3]]  
Output: false

**Coding Challenge 3:**

**Number of Islands**

Description: Count how many islands exist in a 2D binary grid.

**Problem Statement:**

Given an m x n 2D binary grid which represents a map of '1's (land) and '0's (water), return the number of islands. An island is formed by connecting adjacent lands horizontally or vertically. Assume all four edges of the grid are surrounded by water.

**Input Format:**

- grid: List of lists containing '0' or '1'

**Constraints:**

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

**Output Format:**

Return an integer count of islands.

**Tags:**

Matrix, DFS, BFS, Union Find, Flood Fill

**Test Cases:**

Test Case 1:  
Input: [["1","1","1","1","0"],["1","1","0","1","0"],["1","1","0","0","0"],["0","0","0","0","0"]]  
Output: 1

Test Case 2:  
Input: [["1","1","0","0","0"],["1","1","0","0","0"],["0","0","1","0","0"],["0","0","0","1","1"]]  
Output: 3

Test Case 3:  
Input: [["1"]]  
Output: 1

Test Case 4:  
Input: [["0"]]  
Output: 0

Test Case 5:  
Input: [["1","0","1","0"],["0","1","0","1"],["1","0","1","0"],["0","1","0","1"]]  
Output: 8