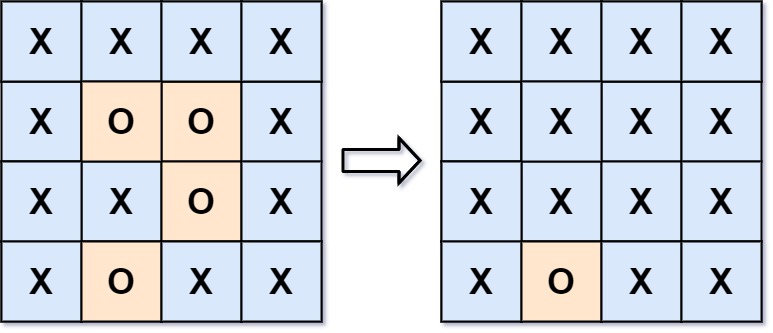
Given an m x n matrix board containing 'X' and 'O', *capture all regions that are 4-directionally surrounded by* 'X'.

A region is **captured** by flipping all 'O's into 'X's in that surrounded region.

**Example 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:** Surrounded regions should not be on the border, which means that any 'O' on the border of the board are not flipped to 'X'. Any 'O' that is not on the border and it is not connected to an 'O' on the border will be flipped to 'X'. Two cells are connected if they are adjacent cells connected horizontally or vertically.

**Example 2:**

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

**Output:** [["X"]]

**Solution:**

public class Solution {

public void solve(char[][] board) {

if (board == null || board.length == 0 || board[0].length == 0) return;

int m = board.length, n = board[0].length;

Queue<int[]> queue = new LinkedList<>();

// Step 1: Get all 'O' on edges first

for (int i = 0; i < m; i++) {

if (board[i][0] == 'O') addPoint(board, queue, i, 0);

if (board[i][n-1] == 'O') addPoint(board, queue, i, n-1);

}

for (int j = 0; j < n; j++) {

if (board[0][j] == 'O') addPoint(board, queue, 0, j);

if (board[m-1][j] == 'O') addPoint(board, queue, m-1, j);

}

// Step 2: BFS for 'O' and put them to '+'

while (!queue.isEmpty()) {

int[] p = queue.poll();

int x = p[0], y = p[1];

if (x-1 >= 0 && board[x-1][y] == 'O') addPoint(board, queue, x-1, y);

if (x+1 <= m-1 && board[x+1][y] == 'O') addPoint(board, queue, x+1, y);

if (y-1 >= 0 && board[x][y-1] == 'O') addPoint(board, queue, x, y-1);

if (y+1 <= n-1 && board[x][y+1] == 'O') addPoint(board, queue, x, y+1);

}

// Step 3: Put all '+' to 'O' and 'O' to 'X'

for (int i = 0; i < m; i++) {

for (int j = 0; j < n; j++) {

if (board[i][j] == 'O') board[i][j] = 'X';

if (board[i][j] == '+') board[i][j] = 'O';

}

}

}

// Change 'O' to '+' and add the point to queue

private void addPoint(char[][] board, Queue queue, int i, int j) {

board[i][j] = '+';

int arr1[] = new int[2];

arr1[0] = i;

arr1[1] = j;

queue.add(arr1);

}

}