Problem Statement :

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.

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: Notice that an 'O' should not be flipped if:
- It is on the border, or
- It is adjacent to an 'O' that should not be flipped.
The bottom 'O' is on the border, so it is not flipped.
The other three 'O' form a surrounded region, so they are flipped.

My Approach: Mohan Gundluri

Using dfs:

Traversing the board when I get "O" I call the dfs function to check the surrounding elements and call the dfs for those elements and changing the "O" -> "Z".

Traversing the board if I get "Z" I change it to "O" for others I changed it to "X".

Using bfs:

Traversing the borders in the board and when I get "O" I will appending the "O"s index in the queue.

Then I'm poping a element's index from queue and changing the value to "Z", then appending the index of surrounding element in the queue, Continuing the process until queue empty.

Traversing the board if I get "Z" I change it to "O" for others I changed it to "X".

My code :

Mohan code

Sonali Gudey and **Deva Rithish Punna** also did the same as I did.

Sonali's code: Sonali Gudey

Sonali`s code

Deva Rithish Punna`s code : Rithish Punna

Rithish`s code

Rohit's code:

I have gone with Rohith code this is also same as our approach but the difference is we appending all the surrounding elements in to the queue but he checking the value of that element, and append the element if value is "O". This makes no effect in the time complexity but affects memory(as we append all elements.)

Rohith's code: Rohith Boodireddy

Rohith`s code

Conclusion:

I have observed my peers code and compared my code for this I have learned how to reduce the space complexity, for this problem and tried it and I got some thoughts after going through my peers codes and I am working on it to get much efficient code for the problem.