Problem 130: Surrounded Regions

Problem Information

Difficulty: Medium
Acceptance Rate: 44.17%
Paid Only: No
Tags: Array, Depth-First Search, Breadth-First Search, Union Find, Matrix

Problem Description

You are given an `m x n` matrix `board` containing **letters** `'X'` and `'O'`, **capture regions** that are **surrounded** :

* **Connect** : A cell is connected to adjacent cells horizontally or vertically. * **Region** : To form a region **connect every** `'O'` cell. * **Surround** : The region is surrounded with `'X'` cells if you can **connect the region** with `'X'` cells and none of the region cells are on the edge of the `board`.

To capture a **surrounded region** , replace all `'O'`s with `'X'`s **in- place** within the original board. You do not need to return anything.

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:

![](https://assets.leetcode.com/uploads/2021/02/19/xogrid.jpg)

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

Example 2:

Input: board = [["X"]]

Output: [["X"]]

Constraints:

* `m == board.length` * `n == board[i].length` * `1 <= m, n <= 200` * `board[i][j]` is `'X'` or `'O'`.

Code Snippets

C++:

```
class Solution {
public:
void solve(vector<vector<char>>& board) {

}
};
```

Java:

```
class Solution {
public void solve(char[][] board) {

}
}
```

Python3:

```
class Solution:
def solve(self, board: List[List[str]]) -> None:
"""
Do not return anything, modify board in-place instead.
"""
```