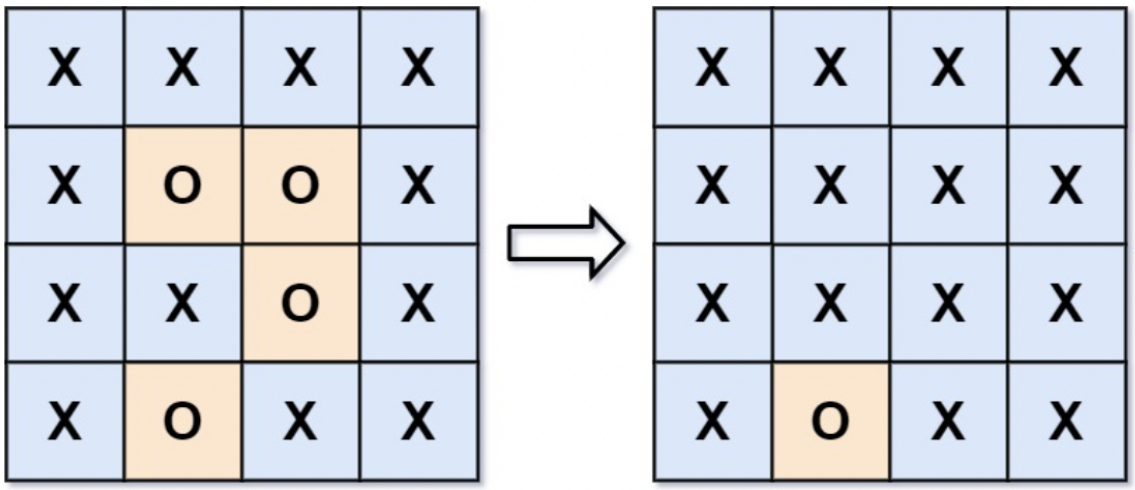
Python Assignment

Peer Learning Document

Problem: Surrounded Regions

Statement:

Given an m*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.

Mohan Gundluri's Approach

- Any 'O' that is not on the border and it is not connected to an 'O' on the border will be flipped to 'X'.
- Firstly, find O's which are on the boundaries of the matrix and do DFS to find another O which is connected.
- While doing the search, temporarily change O's to another character 'Z'.
- After making these changes we get the desired output.

Observation:

Depth-first search is an algorithm for searching in data structures. The algorithm starts at some arbitrary node and explores as far as possible along each branch before backtracking.

Implementation of DFS
In order to determine which "O" entries are connected to it and therefore cannot be captured by "X" because one of its sides is occupied by "O," we traversed the board's borders and performed DFS to all of the "O" entries in the borders. We then marked all of these types of entries to "Y" and

transformed them to "O," whereas other entries were made to "X" because they could be captured.

2. Time and Space Complexity:

Depth First Search Time Complexity : O(m*n)
Space Complexity : O(m*n), where m,n are rows and columns on board.

Mohan's code: https://github.com/mohangundluri2510/Python-Assign

Rohith Boodireddy's Approach

- Traversing every element on the board, and if we come across 'O' we move it to dfs and change the connected 'O'.
- Traversing every element on the board, and if we come across '$' we make it 'O' else we make it 'X'.
- After the changes we do, we get our required output.

1. Observation:

DFS is a recursive method that searches every vertex in a graph or tree data structure. Beginning with the first node of graph G, the depth-first search (DFS) algorithm digs down until it reaches the target node, also known as the node with no children.

The DFS method can be implemented using a stack data structure due to its recursive nature.

2. Time and Space Complexity:

Depth First Search Time Complexity : O(m*n)
Space Complexity : O(m*n) → Stack space, where m,n are rows and columns on board.

- Rohith's code doesn't traverse the entire board and just iterates through the borders, which saves some time when we consider the problem for larger input values.

Rohith's code: https://github.com/Rohith131102/Python_Assignment

Rithish Punna's Approach

- Find all the Os in the borders.
- Do a Depth First Search from the above Os and turn them to Y.
- Then Make the other O's Xs and the Y's Os.

1. Observation:

We notice that all the O's form islands or groups and X's do so too. So now we need to make all the O's present to X's that are surrounded in four directions by X islands.
Now we notice that the O's that are to the side of the square can not obviously be surrounded by X's so all the O islands that have one or more Os that border the side of the square can not be surrounded by Xs however the other islands will be turned to X.

2. Time and Space Complexity:

Depth First Search Time Complexity : O(m*n)
Space Complexity : O(m*n), where m,n are rows and columns on board.

Rithish's code: https://github.com/rithish1126/Python-Assignment

Comparison:

- My peers and I have done the problem almost with the same approach. We started by traversing all the boundaries of the board.
- That means we've done a depth first search on the borders i.e., top and bottom row cells and also for the first and last column cells.
- As we traverse the cells we marked the cells that have values 'O' to some random character.
- After this is done, we've traversed through the grid and changed the cell value to 'X' if its value is 'O' and to 'O' if it has a random character.

Conclusion:

- From this we have learned different approaches to do a single problem.
- I've learnt that we can do this problem in two different ways: bfs and dfs.
- Also there is a solution where there is less time complexity.