题目：

给定一个二维的矩阵，包含 `'X'` 和 `'O'`（**字母 O**）。

找到所有被 `'X'` 围绕的区域，并将这些区域里所有的 `'O'` 用 `'X'` 填充。

示例:

```
X X X X
X O O X
X X O X
X O X X
```

运行你的函数后，矩阵变为：

```
X X X X
X X X X
X X X X
X O X X
```

解释:

被围绕的区间不会存在于边界上，换句话说，任何边界上的 `'O'` 都不会被填充为 `'X'`。 任何不在边界上，或不与边界上的 `'O'` 相连的 `'O'` 最终都会被填充为 `'X'`。如果两个元素在水平或垂直方向相邻，则称它们是“相连”的。

思路：

题目中有两种元素"X"和“O”：

元素"O"还分两种: 一：被X包围的 二：没有被字母X包围的

根据题意：任何边界上的O都不会被填充为X，===》所有不被包围的O都直接或间接与边界上的O相连，由此判断O是否在边界上。

具体说：

对于每一个边界上的O，以它为起点，标记所有与它直接或间接相连的字母O

最后遍历矩阵，对于每一个字母：

如果字母被标记过，则该字母没有被字母X包围的字母O，将其还原为字母O

如果字母没有被标记过，则该字母为被字母X包围的字母O，将其修改为字母X

```python
def solve(self, board):
        """
        :type board: List[List[str]]
        :rtype: None Do not return anything, modify board in-place instead.
        """
        if not board:
            return

        n, m = len(board), len(board[0])

        # 标记边界上的O及与其直接或间接相连接的字母O
        def dfs(x, y):
            # 递归终止条件
            if not 0 <= x < n or not 0 <= y < m or board[x][y] != 'O':
                return
            # 在矩阵内，且board[x][y]==‘O’，就将其标记为‘A’，并向其上下左右扩散
            board[x][y] = "A"
            dfs(x+1, y)
            dfs(x-1, y)
            dfs(x, y+1)
            dfs(x, y-1)


        # 递归上下两个边界的“O”
```

```
        for i in range(n):
            dfs(i, 0)
            dfs(i, m - 1)

        # 递归左右两个边界的“O”
        for i in range(m - 1):
            dfs(0 ,i)
            dfs(n-1, i)


        # 遍历矩阵：如果是被标记过，则将其还原为“O”，如果没有被标记过，则将其改为“X”
        for i in range(n):
            for j in range(m):
                if board[i][j] == "A":
                    board[i][j] = "O"
                elif board[i][j] == "O":
                    board[i][j] = "X"
```

时间复杂度：O（n * m）其中n和m是矩阵的行数和列数， DFS，每个点至多只会被标记一次

空间复杂度：O（n * m）其中n和m是矩阵的行数和列数，主要是DFS栈的开销

BFS:

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

        n, m = len(board), len(board[0])
         que = collections.deque()
         for i in range(n):
            if board[i][0] == "O":
                que.append((i,0))
            if board[i][m-1] == "O":
                que.append((i, m-1))
         for i in range(m-1):
            if board[0][i] == "O":
                que.append((0, i))
            if board[n-1][i] == "O":
                que.append((n-1, i))

         while que:
            x, y = que.popleft()
            board[x][y] = "A"
            for mx, my in [(x-1, y), (x+1, y), (x, y-1), (x, y+1)]:
                if 0 <= mx < n and 0 <= my < m and board[mx][my] == "O":
                    que.append((mx, my))

         for i in range(n):
            for j in range(m):
                if board[i][j] == "A":
```

```
                    board[i][j] = "O"
                elif board[i][j] == "O":
                    board[i][j] = "X"
```

时间复杂度：O(n×m)，其中 n 和 m 分别为矩阵的行数和列数。广度优先搜索过程中，每一个点至多只会被标记一次。

空间复杂度：O(n×m)，其中 n 和 m 分别为矩阵的行数和列数。主要为广度优先搜索的队列的开销。