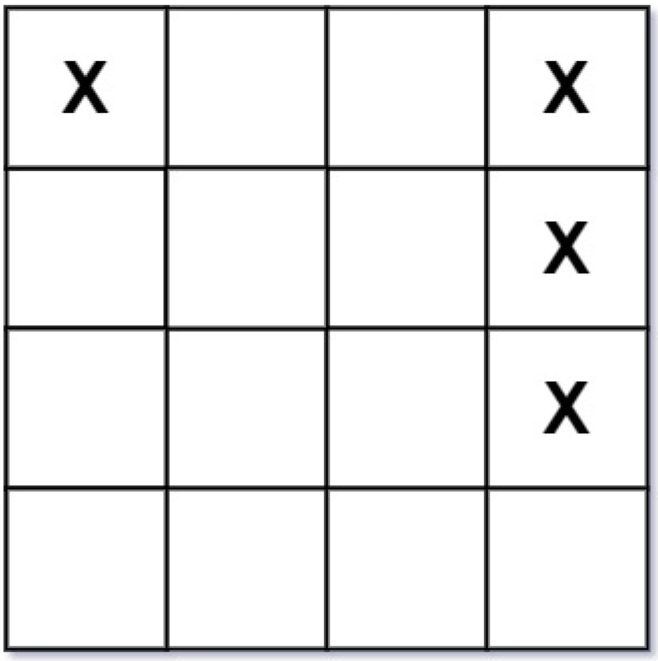
419. Battleships in a Board

Given an `m x n` matrix `board` where each cell is a battleship `'X'` or empty `'.'`, return *the number of the **battleships** on* `board`.

Battleships can only be placed horizontally or vertically on `board`. In other words, they can only be made of the shape `1 x k` (`1` row, `k` columns) or `k x 1` (`k` rows, `1` column), where `k` can be of any size. At least one horizontal or vertical cell separates between two battleships (i.e., there are no adjacent battleships).

Example 1:



```
Input: board = [["X",".",".","X"],[".",".",".","X"],[".",".",".","X"]]
Output: 2
```

Example 2:

```
Input: board = [["."]]
Output: 0
```

Constraints:

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

Follow up: Could you do it in one-pass, using only `O(1)` extra memory and without modifying the values `board`?

```
def countBattleships(self, board: List[List[str]]) -> int:
        ans= 0

        for i in range(len(board)):
            for j in range(len(board[0])):
                if board[i][j]=='X':
                    self.dfs(board,i,j)
                    ans+=1
        return ans



    def dfs(self,board,i,j):
        if i<0 or j<0 or i>=len(board) or j>=len(board[0]) or board[i]
[j]=='.':
            return
        board[i][j]='.'
        self.dfs(board,i-1,j)
        self.dfs(board,i+1,j)
        self.dfs(board,i,j+1)
        self.dfs(board,i,j-1)
```

```
def countBattleships(self, board):
        ans = 0
        for i in xrange(len(board)):
            for j in xrange(len(board[i])):
                if board[i][j] == 'X':
                    if i == 0 or board[i-1][j] != 'X':
                        if j == 0 or board[i][j-1] != 'X':
                            ans += 1
        return ans
```