

419. Battleships in a Board

Given an $m \times n$ 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 $1 \times k$ (1 row, k columns) or $k \times 1$ (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:

X			X
			X
			X

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

Example 2:

Input: board = [["."]]

Output: 0

Constraints:

- $m == \text{board.length}$
- $n == \text{board}[i].\text{length}$
- $1 \leq m, n \leq 200$
- `board[i][j]` is either '.' or 'X'.

Solution

To count the number of battleships, we can first observe that each battleship acts the same as an island described in [this problem](#). We can run the solution to [this same problem](#) however it's unnecessary and a much more elegant solution exists.

The problem statement mentions that each battleship will be a $1 \times k$ or $k \times 1$ rectangle and no battleships are adjacent. Instead of counting battleships, we can pick a cell in each battleship that defines it. Let's call this the **leader** of the ship. To solve the problem, we can simply count the number of **leaders**. The total number of **leaders** we count will be the same as the number of battleships.

For each battleship, we will let the **leader** be the leftmost and upmost cell out of all cells in that battleship.

Example

	0	1	2	3	4	5
0	X	X				
1						X
2		X	X	X		X
3						X
4		X				X

In this specific example, there are four battleships with **leaders** (indicated with blue) located at $(0,0)$, $(2,1)$, $(4,1)$, and $(1,5)$.

How can we determine efficiently if a cell is a **leader**?

Since the **leader** is located in the top-left cell of a battleship and no two battleships are adjacent, we can observe that a cell with an 'X' is a **leader** if the cells to the left (`board[i][j-1]`) and above (`board[i-1][j]`) it (if they exist) are not 'X' cells. In addition, cells that are not **leaders** will always have an 'X' cell to the left or above it so they will never be counted. To count the total number of **leaders**, we'll iterate through all cells and count the number of cells that satisfy the condition mentioned above.

Time Complexity

We can check if a cell is a leader in $\mathcal{O}(1)$ and since there are $\mathcal{O}(MN)$ cells, our time complexity is $\mathcal{O}(MN)$.

Time Complexity: $\mathcal{O}(MN)$

Space Complexity

Since we only maintain a counter for the number of leaders, our space complexity is $\mathcal{O}(1)$.

Space Complexity: $\mathcal{O}(1)$

```
class Solution {
public:
    int countBattleships(vector<vector<char>>& board) {
        int ans = 0;
        int m = board.size();
        int n = board[0].size(); // dimensions for board
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (board[i][j] == 'X') {
                    if ((i == 0 || board[i - 1][j] == '.') && (j == 0 || board[i][j - 1] == '.')) {
                        // check if cell is a leader
                        ans++;
                    }
                }
            }
        }
        return ans;
    }
};

class Solution {
public int countBattleships(char[][] board) {
    int ans = 0;
    int m = board.length;
    int n = board[0].length; // dimensions for board
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (board[i][j] == 'X') {
                if ((i == 0 || board[i - 1][j] == '.') && (j == 0 || board[i][j - 1] == '.')) {
                    // check if cell is a leader
                    ans++;
                }
            }
        }
    }
    return ans;
}

}

class Solution:
    def countBattleships(self, board: List[List[str]]) -> int:
        ans = 0
        m = len(board)
        n = len(board[0]) # dimensions for board
        for i in range(m):
            for j in range(n):
                if board[i][j] == "X":
                    if (i == 0 or board[i - 1][j] == ".") and (
                        j == 0 or board[i][j - 1] == "."):
                        # check if cell is a leader
                        ans += 1
        return ans
```