1861. Rotating the Box with a grid

Medium

Topics

Companies

Hint

You are given an `m x n` matrix of characters `box` representing a side-view of a box. Each cell of the box is one of the following:

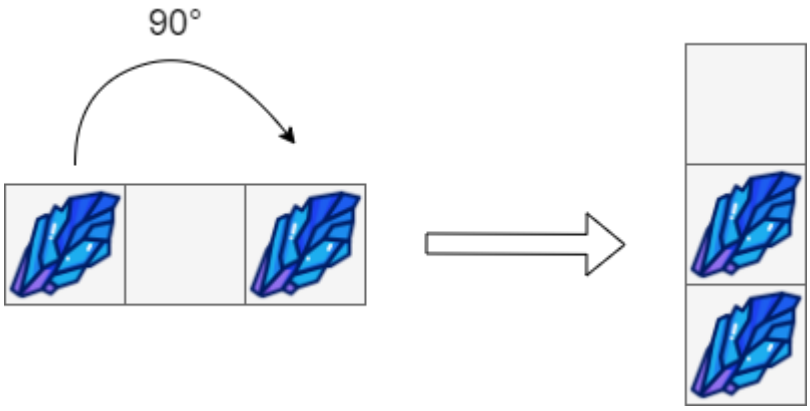
- A stone `'#'`
- A stationary obstacle `'*'`
- Empty `'.'`

The box is rotated **90 degrees clockwise**, causing some of the stones to fall due to gravity. Each stone falls down until it lands on an obstacle, another stone, or the bottom of the box. Gravity **does not** affect the obstacles' positions, and the inertia from the box's rotation **does not** affect the stones' horizontal positions.

It is **guaranteed** that each stone in `box` rests on an obstacle, another stone, or the bottom of the box.

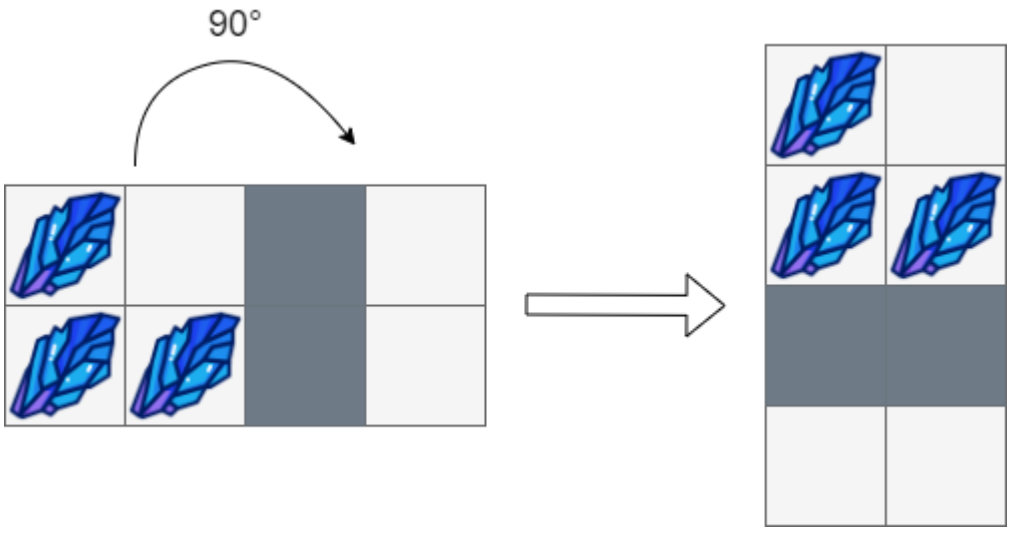
Return *an* `n x m` *matrix representing the box after the rotation described above*.

Example 1:



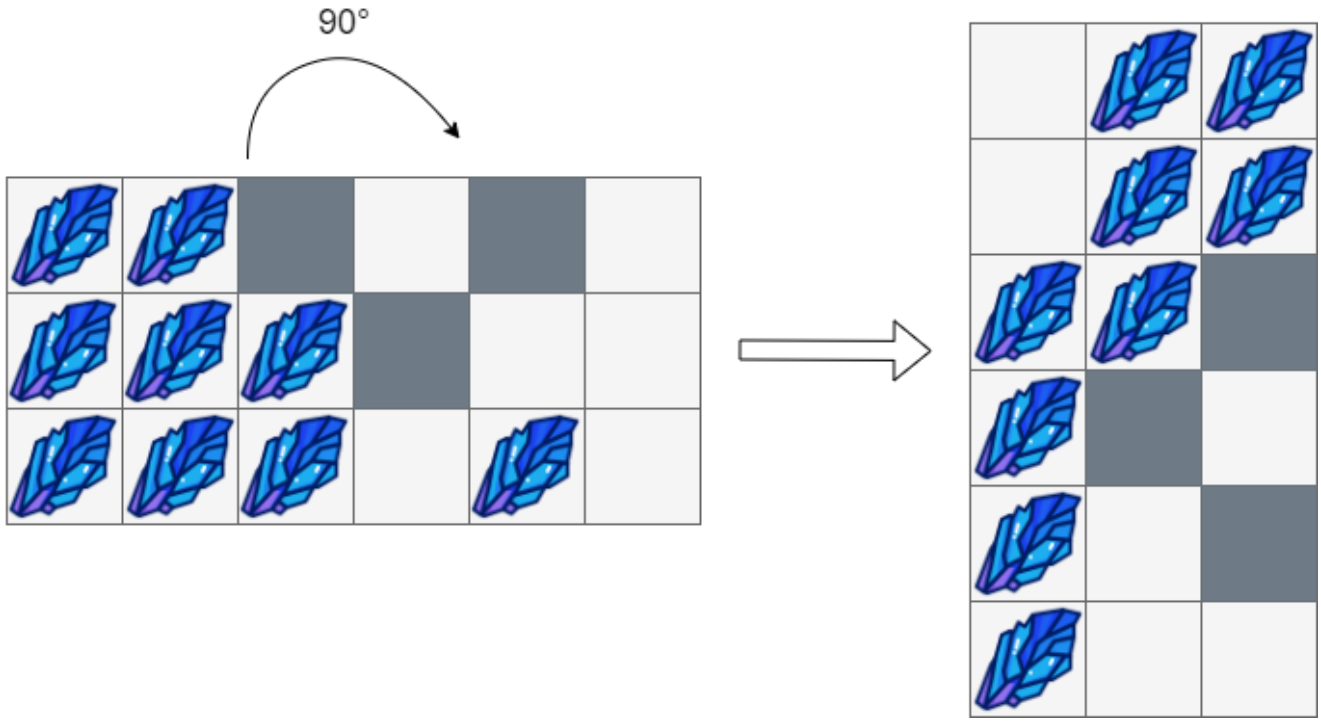
Input: box = [" > ","."," > "]
Output: [["."],
["#"],
["#"]]

Example 2:




```
Input: box = [["#",".","*","."],
["#","#","*","."]]
Output: [["#","."],
["#","#"],
["",""],
[".","."]]
```

Example 3:




```
Input: box = [["#","#","",".","","."],
["#","#","#","*",".","."],
["#","#","#",".","#","."]]
Output: [[".","#","#"],
[".","#","#"],
["#","#","*"],
```

```
["#","*","."],
["#",".","*"],
["#",".","."]]
```

Constraints:

- `m == box.length`
- `n == box[i].length`
- `1 <= m, n <= 500`
- `box[i][j]` is either `'#'`, `'*'`, or `'.'`.

Solution:

```java
class Solution {
    public char[][] rotateTheBox(char[][] box) {
        int m = box.length;     // Number of rows
        int n = box[0].length;  // Number of columns

        // Apply gravity on each row
        for (int i = 0; i < m; i++) {
            int emptyPos = n - 1;  // Start from the rightmost column
            for (int j = n - 1; j >= 0; j--) {
                if (box[i][j] == '*') {
                    // Obstacle, move empty position to one before this
                    emptyPos = j - 1;
                } else if (box[i][j] == '#') {
                    // If it's a stone, move it to the farthest empty
position
                    box[i][j] = '.';
                    box[i][emptyPos] = '#';
                    emptyPos--; // Update the empty position
                }
            }
        }

        // Rotate the box 90 degrees clockwise
        char[][] rotatedBox = new char[n][m];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                rotatedBox[j][m - 1 - i] = box[i][j];
            }
        }
        return rotatedBox;
    }
}
```