Rotating The Box

solved by	Senan
Platform	LeetCode
difficulty	Medium
Serial	1861
tags	Reverse Transpose
language	C++
solved on	@23/11/2024
link	https://leetcode.com/problems/rotating-the-box/description/
Completion	☑️

Intuition

The problem involves simulating the falling of stones (`#`) in a box represented as a 2D grid after rotating it 90 degrees clockwise. The goal is to achieve this efficiently by combining two key operations: handling the stone movement within each row (before rotation) and then rotating the grid.

Approach

1. **Handle stone falling in rows**:
   - For each row, simulate the movement of stones (`#`) toward the right end of the row until they encounter an obstacle (`.`).
   - Use two pointers (`i` and `j`) to track the current position of the stone and the destination respectively, swapping as needed.
2. **Transpose the box**:
   - Rotate the grid 90 degrees clockwise by first transposing the matrix (swapping rows and columns).
3. **Reverse rows**:
   - After transposing, reverse each row of the transposed grid to complete the rotation.
4. **Combine these operations**:
   - First, handle stone falling for all rows.
   - Then perform the transpose and row reversal to achieve the final rotated state.

Complexity

Time Complexity:

- **Stone movement simulation**: **O(mn)**, where m is the number of rows and n is the number of columns, as each cell is visited once during stone movement.
- **Transpose operation**: **O(mn)**, since each element is moved to its transposed position.
- **Row reversal**: **O(mn)**, as each row is reversed.

Overall Time Complexity: **O(mn)**

Space Complexity:

- **Auxiliary space for transposed matrix**: **O(mn)**
- **In-place operations** for stone movement do not require extra space.

Overall Space Complexity: **O(mn)**.

Code

```
class Solution {
    void pushTheObstable(int row, vector<vector<char>> &box) {
        vector<char>& obstacles = box[row];
        int n = obstacles.size();
        int i = n - 1, j = n - 1;
        while (0 <= i) {
            if (obstacles[i] == '.') i--;
            else if (obstacles[i] == '*') j = --i;
            else swap(obstacles[i--], obstacles[j--]);
        }
        return;
    }

    vector<vector<char>> transpose(vector<vector<char>>& box) {
        int m = box.size();
        int n = box[0].size();
        vector<vector<char>> rotatedBox(n, vector<char>(m));
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                rotatedBox[j][i] = box[i][j];
            }
        }
        return rotatedBox;
    }

public:
    vector<vector<char>> rotateTheBox(vector<vector<char>>& box) {
        for (int i = 0; i < box.size(); i++)
            pushTheObstable(i, box);

        auto rotated = transpose(box);

        for (int i = 0; i < rotated.size(); i++)
            reverse(rotated[i].begin(), rotated[i].end());

        return rotated;
    }
};
```