Leetcode - 54

Given an `m x n` matrix, return all elements in spiral order.

📥 Example 1:

```
Input: matrix = [[1,2,3],
                 [4,5,6],
                 [7,8,9]]
Output: [1,2,3,6,9,8,7,4,5]
```

📥 Example 2:

```
Input: matrix = [[1,2,3,4],
                 [5,6,7,8],
                 [9,10,11,12]]
Output: [1,2,3,4,8,12,11,10,9,5,6,7]
```

💡 Approach:

Use four pointers:

- `minr`, `maxr`: top and bottom row boundaries
- `minc`, `maxc`: left and right column boundaries

Traverse in this order:

1. Left → Right
2. Top ⬇️ Bottom
3. Right ⬅️ Left
4. Bottom ⬆ Top

Repeat this in a loop until all elements are visited.

✅ Full C++ Code:

```
class Solution {
public:
    vector<int> spiralOrder(vector<vector<int>>& matrix) {
        if (matrix.empty() || matrix[0].empty()) return {};

        int m = matrix.size();
        int n = matrix[0].size();

        int minc = 0, maxc = n - 1;
        int minr = 0, maxr = m - 1;
        vector<int> v;

        while (minc <= maxc && minr <= maxr) {
            // right
            for (int j = minc; j <= maxc; j++)
                v.push_back(matrix[minr][j]);
            minr++;

            // down
            for (int i = minr; i <= maxr; i++)
                v.push_back(matrix[i][maxc]);
            maxc--;

            // left
            if (minr <= maxr) {
                for (int j = maxc; j >= minc; j--)
                    v.push_back(matrix[maxr][j]);
                maxr--;
            }

            // up
            if (minc <= maxc) {
                for (int i = maxr; i >= minr; i--)
                    v.push_back(matrix[i][minc]);
```

```
            minc++;
          }
        }

        return v;
    }
};
```

📊 Complexity:

- **Time:** O(m × n)
- **Space:** O(1) (excluding output vector)