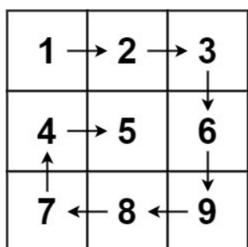


30. Spiral Traversal of a Matrix

Given an $m \times n$ matrix, return all elements of the matrix in spiral order.

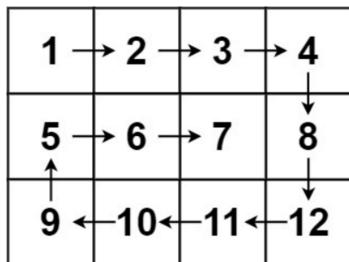
Example 1:



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]

$n \times m \Rightarrow$ print it in spiral order.

1 2 3 4 5 6

20 21 22 23 24 7

19 32 33 34 25 8

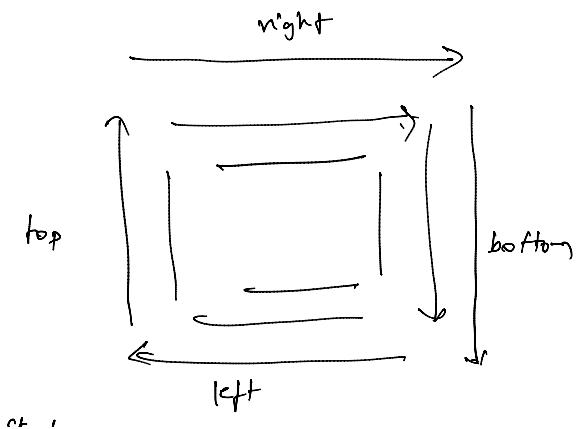
Output: 1 2 3 4 5 6 7 8 ... 36

18 31 36 35 26 9

17 30 29 28 27 10

16 15 14 13 12 11

Optimal Solution:



← left ↓
 1st layer
 right → bottom → left → top

2nd layer
 right → bottom → left → top
 3rd layer
 right → bottom → left

left ← left right → right
 0 1 2 3 4 5

top ↓
 0 1 2 3 4 5 ↓
 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 59 60

X → to avoid repetition (for the top we move down to next index)

							top = 0 bottom = 5
2	19	32	33	34	25	8	
3	18	31	36	35	26	9	left = 0 right = 5

							10
4	17	30	29	28	27	10	
5	16	15	14	13	12	11	right
	left ←						as bottom is done "11" should be ignored so move right back

1st layer printing
 1st printing left → right

for (left → right) } right will print

$a[\text{top}][i]$ } it stay constant

top++; } this will make sure right is performed if it's available.

2nd printing
 top is gone down now we need to print bottom

so we go top → bottom

$\text{for } (i = \text{top} \rightarrow \text{bottom})$
 it stay const $a[\text{top}] [i]$
 $\text{top}++$

3^{rd} printing right \rightarrow left
 $\text{for } (i = \text{right} \rightarrow \text{left})$
 $a[\text{bottom}] [i]$
 $\text{bottom}--$

4^{th} printing top

bottom \rightarrow top

$\text{for } (i = \text{bottom} \rightarrow \text{top})$

$a[i] [\text{left}]$;
 $\text{left}++$;

2nd layer printing

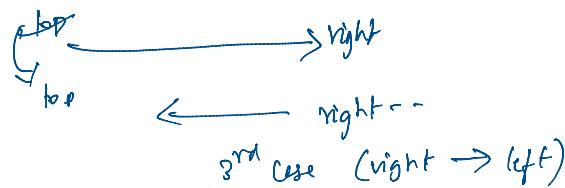
if the left is smaller than right i.e there is column
 that has to be traversed (spiral)
 similarly for top is lesser than bottom, " " "
 $\rightarrow \text{while } (\text{top} \leq \text{bottom} \& \& \text{left} \leq \text{right})$

Now if we just single line it will perform only right



1st case will perform //right.
 now top++

No need to do extra checks



but top is exceeds so check

$\text{if } (\text{top} <= \text{bottom})$

```
1 vector<int> spiralMatrix(vector<vector<int>> &matrix) {
2     int n = matrix.size();
3     int m = matrix[0].size();
4
5     int left = 0, right = m - 1;
6     int top = 0, bottom = n - 1;
7
8     vector<int> ans;
9     while (top <= bottom && left <= right) {
10        // right
11        for (int i = left; i <= right; i++) {
12            ans.push_back(matrix[top][i]);
13        }
14        top++;
15
16        // bottom
17        for (int i = top; i <= bottom; i++) {
18            ans.push_back(matrix[i][right]);
19        }
20        right--;
21
22        if (top <= bottom) {
23            // left
24            for (int i = right; i >= left; i--) {
25                ans.push_back(matrix[bottom][i]);
26            }
27            bottom--;
28        }
29        if (left <= right) {
30            // top
31            for (int i = bottom; i >= top; i--) {
32                ans.push_back(matrix[i][left]);
33            }
34            left++;
35        }
36    }
37    return ans;
38}
```

```
i C++ • Autocomplete
1 class Solution {
2 public:
3     vector<int> spiralOrder(vector<vector<int>> &matrix) {
4         int n = matrix.size();
5         int m = matrix[0].size();
6
7         int left = 0, right = m - 1;
8         int top = 0, bottom = n - 1;
9
10        vector<int> ans;
11        while (top <= bottom && left <= right) {
12            // right
13            for (int i = left; i <= right; i++) {
14                ans.push_back(matrix[top][i]);
15            }
16            top++;
17
18            // bottom
19            for (int i = top; i <= bottom; i++) {
20                ans.push_back(matrix[i][right]);
21            }
22            right--;
23
24            if (top <= bottom) {
25                // left
26                for (int i = right; i >= left; i--) {
27                    ans.push_back(matrix[bottom][i]);
28                }
29                bottom--;
30            }
31            if (left <= right) {
32                // top
33                for (int i = bottom; i >= top; i--) {
34                    ans.push_back(matrix[i][left]);
35                }
36                left++;
37            }
38        }
39        return ans;
40    }
41 }
```

$T.C \Rightarrow O(n \times m)$

$S.C \Rightarrow O(n \times m)$