Previous Next

54. Spiral Matrix

Sept. 6, 2017 | 209.2K views

Average Rating: 3.98 (94 votes)

Given a matrix of *m* x *n* elements (*m* rows, *n* columns), return all elements of the matrix in spiral order.

Example 1:

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

Example 2:

```
Input:
[
  [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 1: Simulation

Intuition

Draw the path that the spiral makes. We know that the path should turn clockwise whenever it would go out of bounds or into a cell that was previously visited.

Algorithm

Let the array have R rows and C columns. seen[r][c] denotes that the cell on the$\text{r}$-th row and c-th column was previously visited. Our current position is (r, c), facing direction di, and we want to visit R x C total cells.

As we move through the matrix, our candidate next position is (cr, cc). If the candidate is in the bounds of the matrix and unseen, then it becomes our next position; otherwise, our next position is the one after performing a clockwise turn.

Java | Python | Copy

```python
class Solution(object):
    def spiralOrder(self, matrix):
        if not matrix: return []
        R, C = len(matrix), len(matrix[0])
        seen = [[False] * C for _ in matrix]
        ans = []
        dr = [0, 1, 0, -1]
        dc = [1, 0, -1, 0]
        r = c = di = 0
        for _ in range(R * C):
            ans.append(matrix[r][c])
            seen[r][c] = True
            cr, cc = r + dr[di], c + dc[di]
            if 0 <= cr < R and 0 <= cc < C and not seen[cr][cc]:
                r, c = cr, cc
            else:
                di = (di + 1) % 4
                r, c = r + dr[di], c + dc[di]
        return ans
```

Complexity Analysis

- Time Complexity: $O(N)$, where N is the total number of elements in the input matrix. We add every element in the matrix to our final answer.
- Space Complexity: $O(N)$, the information stored in `seen` and in `ans`.

Approach 2: Layer-by-Layer

Intuition

The answer will be all the elements in clockwise order from the first-outer layer, followed by the elements from the second-outer layer, and so on.

Algorithm

We define the k-th outer layer of a matrix as all elements that have minimum distance to some border equal to k. For example, the following matrix has all elements in the first-outer layer equal to 1, all elements in the second-outer layer equal to 2, and all elements in the third-outer layer equal to 3.

```
[[1, 1, 1, 1, 1, 1, 1],
 [1, 2, 2, 2, 2, 2, 1],
 [1, 2, 3, 3, 3, 2, 1],
 [1, 2, 2, 2, 2, 2, 1],
 [1, 1, 1, 1, 1, 1, 1]]
```

For each outer layer, we want to iterate through its elements in clockwise order starting from the top left corner. Suppose the current outer layer has top-left coordinates $(\text{r1}, \text{c1})$ and bottom-right coordinates $(\text{r2}, \text{c2})$.

Then, the top row is the set of elements $(\text{r1}, \text{c})$ for $\text{c} = \text{c1},\dots,\text{c2}$, in that order. The rest of the right side is the set of elements $(\text{r}, \text{c2})$ for $\text{r} = \text{r1}+1,\dots,\text{r2}$, in that order. Then, if there are four sides to this layer (ie., $\text{r1} < \text{r2}$ and $\text{c1} < \text{c2}$), we iterate through the bottom side and left side as shown in the solutions below.

```
[ [ 1, 1, 1, 1, 1, 1 ],     top: c from c1 ... c2
  [ 1, 2, 2, 2, 2, 1 ],     right: r from r1+1 ... r2
  [ 1, 2, 3, 3, 2, 1 ],     bottom: c from c2+1 ... c1+1
  [ 1, 2, 2, 2, 2, 1 ],     left: r from r2+1 ... r1+1
  [ 1, 1, 1, 1, 1, 1 ] ]
```

Java | Python | Copy

```python
class Solution(object):
    def spiralOrder(self, matrix):
        def spiral_coords(r1, c1, r2, c2):
            for c in range(c1, c2 + 1):
                yield r1, c
            for r in range(r1 + 1, r2 + 1):
                yield r, c2
            if r1 < r2 and c1 < c2:
                for c in range(c2 - 1, c1, -1):
                    yield r2, c
                for r in range(r2, r1, -1):
                    yield r, c1

        if not matrix: return []
        ans = []
        r1, r2 = 0, len(matrix) - 1
        c1, c2 = 0, len(matrix[0]) - 1
        while r1 <= r2 and c1 <= c2:
            for r, c in spiral_coords(r1, c1, r2, c2):
                ans.append(matrix[r][c])
            r1 += 1; r2 -= 1
            c1 += 1; c2 -= 1
        return ans
```

Complexity Analysis

- Time Complexity: $O(N)$, where N is the total number of elements in the input matrix. We add every element in the matrix to our final answer.
- Space Complexity:
  - $O(1)$ without considering the output array, since we don't use any additional data structures for our computations.
  - $O(N)$ if the output array is taken into account.

Rate this article:

Previous Next

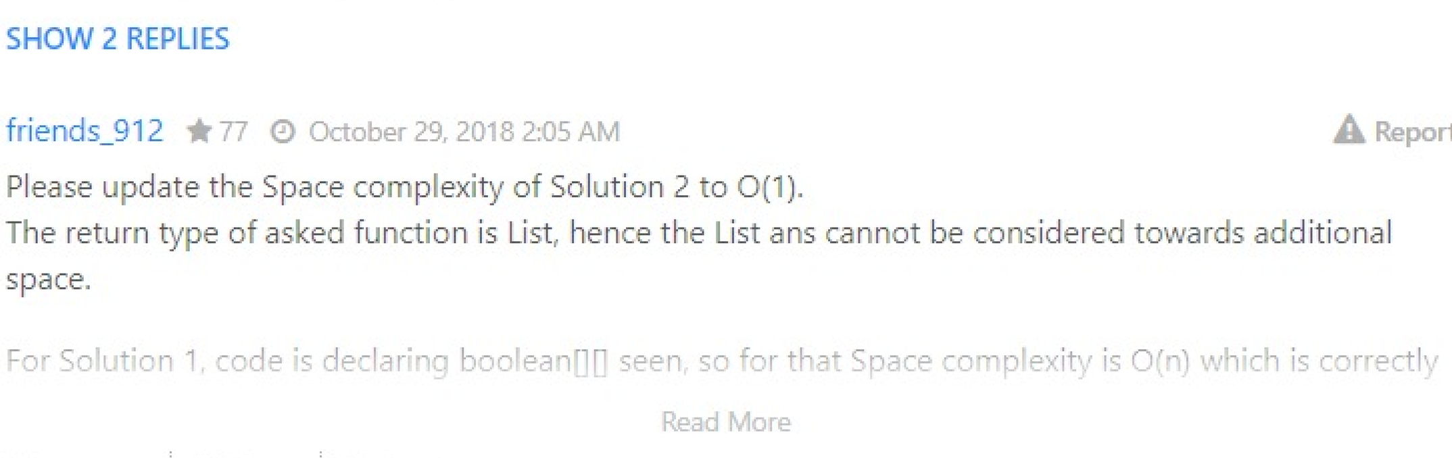
Comments: 65

Sort By

Type comment here... (Markdown is supported)

Preview Post

thawsitt9 ★ 100 August 9, 2018 3:50 AM



100 Share Reply

SHOW 2 REPLIES

friends_912 ★ 77 October 29, 2018 2:05 AM Report

Please update the Space complexity of Solution 2 to O(1).
The return type of asked function is List, hence the List ans cannot be considered towards additional space.

For Solution 1, code is declaring boolean[][] seen, so for that Space complexity is O(n) which is correctly

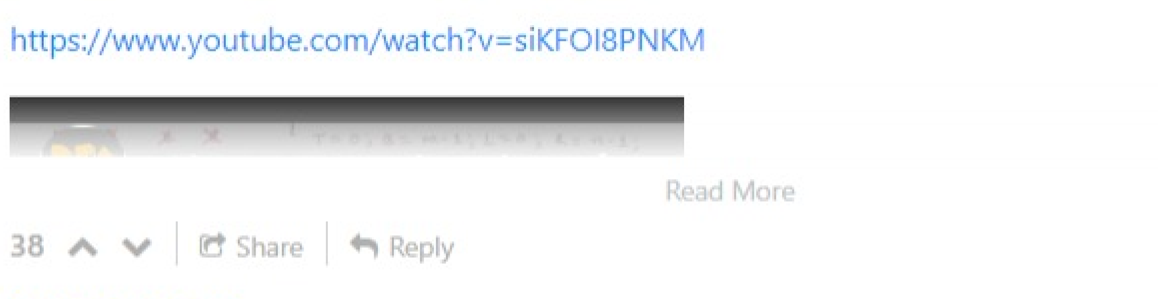
Read More

71 Share Reply

SHOW 6 REPLIES

vegito2002 ★ 1261 February 9, 2018 4:19 AM

Shouldn't the picture for Approach#2 be like this?



42 Share Reply

SHOW 1 REPLY

shadowmaster123 ★ 82 May 5, 2019 8:25 AM

For anyone having trouble understanding, check this video, its not mine, but thought I'd share.

https://www.youtube.com/watch?v=siKFOI8PNKM

Read More

38 Share Reply

SHOW 3 REPLIES

softwareshortcut ★ 424 May 1, 2019 11:04 PM

I think it's the ultimate exercise to train with arrays and indices. If you're dyslexic like myself, good luck! :)

28 Share Reply

SHOW 3 REPLIES

pdne91 ★ 47 January 8, 2019 10:02 AM Report

Space complexity usually doesn't include the space required to return the results. I would argue that space complexity for approach 1 is O(N) for the auxiliary array of booleans, and O(1) for approach 2.

11 Share Reply

Dr_Sean ★ 535 April 26, 2019 1:03 AM

Python3 solution with explanation (No change is made in the matrix):

```
class Solution(object):
    def spiralOrder(self, matrix):
        if len(matrix) == 0: return []
```

Read More

8 Share Reply

yaoct ★ 18 January 8, 2019 9:29 PM

```
class Solution {
    public List<Integer> spiralOrder(int[][] matrix) {
        List<Integer> l=new ArrayList<>();
        if(matrix.length==0) return l;
```

Read More

6 Share Reply

pradeep11 ★ 3 September 20, 2019 4:40 AM

Can someone please explain, how the direction array values interpreted in the first approach? Thanks

2 Share Reply

SHOW 1 REPLY

Bungalow12 ★ 3 June 12, 2018 11:44 PM Report

Here is my C# solution. I feel like it is clean and easy to read. Also performed better than 89.41% of C# solutions. Hopefully this is helpful.

```
public class Solution {
    public IList<int> SpiralOrder(int[,] matrix)
```

Read More

2 Share Reply

‹ 1 2 3 4 5 6 7 ›