Approach #2: Layer-by-Layer [Accepted]

**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 (r1, c1) and bottom-right coordinates (r2, c2).

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

Note that r1 = rowStart, r2 = rowEnd, c1 = colStart, c2 = colEnd.

**Complexity Analysis**

* Time Complexity: O(N)*O*(*N*), where N*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)*O*(*N*), the information stored in ans.

