



// My First Solution

class Solution:

def spiralOrder(self, matrix: List[List[int]]) -> List[int]:

# Get Matrix dimensions

M = len(matrix) # rows

N = len(matrix[0]) # cols

T = M\*N # Total elements

# output array

output = []

# set edges

ue = 0 # upper edge

de = M-1 # lower edge

le = 0 # left edge

re = N-1 #right edge

# starting indices

r,c = 0,0

# using simulation, exactly what question asks you to do

# complete one round, shift inside one block, repeat

while T != 0:

# condition for going right

if (c == le and r ==ue):

while c <= re: # stopping condition

output.append(matrix[r][c])

c += 1

T -= 1

c -= 1 # bring c back in matrix range

ue += 1

# condition for going down

elif (c == re and r!=de):

r += 1

while r <= de:

output.append(matrix[r][c])

r += 1

T -= 1

r -= 1

re -= 1

# condition for going left

elif (r == de and c!= le):

c -= 1

while c>= le:

output.append(matrix[r][c])

c -= 1

T -= 1

c += 1

de -= 1

# condition for going up

elif (c == le and r!= ue):

r -= 1

while r >= ue:

output.append(matrix[r][c])

r -= 1

T -= 1

r += 1

le += 1

c += 1

return output

result ->

**23 / 23** test cases passed.

Runtime: **24 ms (98%)**

Memory Usage: **14 MB (82%)**